МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до магістерської роботи
магістра
(ступінь вищої освіти)

на тему **КОМП'ЮТЕРНА СИСТЕМА МОДЕЛЮВАННЯ ПОВЕДІНКИ ЛЮДЕЙ В ЕКСТРЕМАЛЬНИХ СИТУАЦІЯХ**

Виконав: студент 2 курсу, групи КНТ-513м
спеціальності
123 «Комп'ютерна інженерія»
(код і найменування спеціальності)
Освітня програма (спеціалізація)
«Комп'ютерні системи та мережі»
ШУМОВ В.А.
(ПРІЗВИЩЕ та ініціали)
Керівник СКРУПСЬКИЙ С. Ю.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОЗІНА Г.Л.
(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) «Комп'ютерна системи та мережі»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.
_____
"____" _____ 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

ШУМОВ Володимир Андрійович
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система моделювання поведінки людей в екстремальних ситуаціях.
керівник проєкту (роботи) к. т. н., доцент, СКРУПСЬКИЙ Степан Юрійович
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від "18" жовтня 2024 року № 149
2. Строк подання студентом проєкту (роботи) 10 грудня 2024 року
3. Вихідні дані до проєкту (роботи) існуючі методи та алгоритми моделювання поведінки людей, моделювання системи за допомогою UML, використання MS Visio.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області;
2) Проєктування системи;
3) Реалізація системи;
4) Дослідження розробленої системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ С.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів розробки поведінки людей в екстримальних ситуаціях, перегляд отриманих результатів, виділення недоліків та переваг.	21.10.2024 р.	
2	Встановлення та налаштування програмного забезпечення	24.10.2024 р.	
3	Аналіз технології UML	25.10.2024 р.	
4	Розробка програмних компонентів системи	12.11.2024 р.	
5	Дослідження розробленої системи	24.11.2024 р.	
6	Оформлення отриманих результатів у ПЗ	02.12.2024 р.	
7	Оформлення графічного матеріалу	06.12.2024 р.	
8	Оформлення допоміжного матеріалу	09.12.2024 р.	

Студент(ка) _____ **Володимир ШУМОВ**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Степан СКРУПСЬКИЙ**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 72 с., 41 рис., 5 лістингів, 1 табл., 11 джерел.

ВИМОГИ, ІНФОРМАЦІЙНА СИСТЕМА, КОМП'ЮТЕРНА СИСТЕМА, МОДЕЛЮВАННЯ ПОВЕДІНКИ, ПРОЄКТУВАННЯ

Об'єкт розробки – комп'ютерна система моделювання поведінки людей в екстримальних ситуаціях.

Ціль даної магістерської роботи є дослідження та розробка спеціалізованих комп'ютерної системи з використанням методів об'єктно-орієнтованого аналізу і моделювання.

Завдання магістерської роботи – розробити модель для проєктування комп'ютерної системи за допомогою мови моделювання UML.

Проєкт складається з 4 розділів.

Перший розділ містить огляд загальної теоретичної інформації про проєктування та його підходи, аналіз предметної області

Другий розділ здійснює опис етапів моделювання системи за допомогою мови UML, проєктування інтерфейсу користувача.

Третій розділ описує реалізація системи моделювання і наведені приклади програмного коду.

Четвертий розділ має зазначені методи дослідження систем та його результати.

Результатом виконання магістерської роботи є створення моделі системи моделювання поведінки людей в екстримальних ситуаціях та модель реалізації інтерфейсу користувача, що автоматизує та допомагає пришвидшити процес евакуації натовпу в екстремальній ситуації.

ABSTRACT

PP: 72 p., 41 figures, 5 listings, 1 table. 11 sources.

REQUIREMENTS, INFORMATION SYSTEM, COMPUTER SYSTEM, BEHAVIOR MODELING, DESIGN

The project consists of 4 sectionsThe object of development is a computer system for modeling the behavior of people in extreme situations.

The goal of this thesis is research and development of specialized computer systems using methods of object-oriented analysis and modeling.

The task of the thesis is to develop a model for designing a computer system using the UML modeling language.

The first section contains an overview of general theoretical information about design and its approaches, analysis of the subject area

The second section describes the stages of system modeling using the UML language, user interface design.

The third section describes the implementation of the simulation system and provides examples of the software code.

The fourth section has specified methods of system exploration and its results.

The result of the diploma work is the creation of a model of the system for modeling people's behavior in extreme situations and a model of the implementation of the user interface that automates and helps speed up the process of evacuating the crowd in an extreme situation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АІС – автоматизована інформаційна система

БД – база даних;

НП – надзвичайні події;

ЕС – екстримальны ситуації;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології

UML – Unified Modeling Language, уніфікована мова моделювання;

GUI – Graphical User Interface, графічний користувацький інтерфейс;

ЕОМ – електронно-обчислювальна машина;

WIMP – (windows, icons, menus, pointers) – вікна, іконки, меню, покажчики

ВСТУП

Сьогодні інформаційні технології відіграють важливу роль у розвитку всіх сфер життя. Без використання сучасних інформаційних технологій був би неможливий стрімкий та активний розвиток сучасного світу. Однак цей стрімкий рух супроводжується величезними обсягами інформації, які фізично не можуть бути сприйняті людиною. Як наслідок, багато галузей промисловості переходять на автоматизовані системи для підвищення продуктивності та зменшення впливу людського фактору.

Тому безпека великої кількості людей у закритих приміщеннях є актуальним питанням, оскільки людські помилки в багатьох екстремальних ситуаціях можуть призвести до численних жертв. За статистикою, більшість аварій є техногенними: у 2011-2016 роках в Україні було зареєстровано 867 екстремальних ситуацій [1]. Темпи зростання кількості техногенних аварій та катастроф у 3 рази перевищують темпи зростання кількості стихійних лих. Ці факти підтверджують нагальну необхідність вирішення проблеми екстреної евакуації.

Це було досягнуто завдяки зростанню автоматизації розробки на основі проєктів вчених на базі комп'ютерних технологій. На початковому етапі проєктуванні будівлі можна не тільки виявити вогнища скупчення людей під час екстреної евакуації, але і передбачити додаткові виходи і збільшення пропускної здатності коридорів, наприклад, при проєктуванні багатоповерхового комплексу можна змоделювати потік людей для усунення вогнищ, орієнтуючись тільки на кількість людей, які можуть увійти в будівлю або на певний поверх, а також перетворення потоку людей у модель.

Тому доцільно розробити систему моделювання руху людських потоків з урахуванням вищезазначеного. Це дозволить чітко визначити безпечні шляхи евакуації та розрахувати час, необхідний для евакуації всіх працівників та відвідувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Коли люди перебувають у натовпі, на їхню поведінку, емоції та готовність діяти впливають фізичні та емоційні зв'язки. Розуміння цих зв'язків може допомогти людям впоратися з панікою, викликаною, наприклад, терористичним нападом. Поведінка натовпу важлива в багатьох екстремальних ситуаціях, особливо коли натовп дуже щільний. Паніка і плутанина в натовпі можуть призвести до сотень поранень і навіть смертей.

Фундаментальні дослідження і громадська безпека вимагають глибокого розуміння поведінки натовпу в багатьох різних сферах. Дослідження істориків і соціальних психологів показали, що великі натовпи перебувають під впливом окремих людей. Природа цих впливів може допомогти нам зрозуміти альтруїстичну і корисну поведінку і, навпаки, агресивну і деструктивну поведінку. Це дослідження дає можливість глибше вивчити поведінку натовпу. Кількісний аналіз, обчислювальні методи і теорія систем можуть бути використані для більш детального вивчення цієї теми.

Щоб бути дійсно корисним у екстремальних ситуаціях, всебічний аналіз динаміки людського руху повинен також враховувати емоційну передачу натовпу. Дослідження науковців технологічного університету показало, що поширення страху може змінити колективну поведінку: у 2015 році вони створили комп'ютерну симуляцію громадського простору з сотнями дітей, дорослих і охоронців, які спрямовували людей до виходів. Симуляція показала, що учасники опинилися в екстримальній ситуації без можливості втечі, що призвело до дуже високого рівня страху і паніки. Таким чином, системи комп'ютерного моделювання можуть бути дуже корисними при проєктуванні евакуації.

Моделювання інформаційних систем вимагає створення інформаційної моделі. Інформаційна модель - це набір спеціальним чином організованих даних

(сигналів, підказок), які відображають найважливіші аспекти об'єкта. Інформаційні моделі можуть бути візуальними, графічними, анімованими, текстовими або табличними. Сюди відносяться всі види діаграм, графіків, таблиць, схем, малюнків та анімації, створених на комп'ютері.

Імітаційне моделювання є інструментом експериментального дослідження складних систем і включає методологію моделювання систем, алгоритмічні методи, програмну реалізацію імітаторів, планування і проведення експериментів з імітаційними моделями на комп'ютерах, обробку даних і аналіз результатів, результати аналіз, серед іншого [2].

Існує три підходи до імітаційного моделювання: системна динаміка, дискретно-подієвий та агентний.

Системна динаміка - це принципи моделювання та дослідження систем, що означуєьбся петлями зворотного зв'язку в складних причинно-наслідкових залежностях між параметрами. Математично ці системи відтворюються системами диференціальних рівнянь скорочених до форм Коші. Системна динаміка абстрагується від окремих об'єктів та подій і зосереджується на політиці, що керує цими процесами, припускаючи «цілісний» погляд на процес.

Дискретне подієве моделювання - це процес кодування поведінки складної системи в упорядковану послідовність чітко визначених подій. У цьому контексті подія - це конкретна зміна стану системи в певний момент часу.

В агентному моделюванні система моделюється як сукупність автономних об'єктів, що приймають рішення, які називаються агентами. Кожен агент індивідуально приймає рішення на основі набору правил. На найпростішому рівні агентна модель складається з системи агентів і взаємозв'язків між ними. Навіть прості агентні моделі можуть демонструвати складну поведінку і надавати цінну інформацію про динаміку реальних систем, які вони моделюють.

Уніфікована мова моделювання (UML) - це стандартна нотація для візуального моделювання програмних систем [2].

Вона підтримується багатьма об'єктно-орієнтованими інструментальними системами. Незалежно від методології, що використовується при розробці

проєкту, UML описує класи, об'єкти та компоненти. Мови графічного візуального моделювання мають свою власну нотацію, прийнятну нотацію (Acceptable Notation) [3]. Ця нотація забезпечує семантику мови і є способом уніфікації нотацій візуального моделювання, забезпечуючи комплексне представлення системи, яке є відносно простим і зрозумілим для людини.

У процесі системи з різних точок зору, а їхні різноманітні представлення відображаються у вигляді діаграм.

Діаграма - це графічне представлення набору елементів. Діаграма зазвичай має вигляд графа з вершинами (сутностями) і ребрами (зв'язками).

Поведінка системи описується за допомогою функціональної моделі, яка показує передумови системи (варіанти використання), оточення системи (актори, дійові особи) і взаємозв'язки між ними (діаграми варіантів використання).

Діаграма варіантів використання - це діаграма, що показує взаємозв'язок між акторами та варіантами використання. Ці діаграми можна використовувати для наступних цілей:

– визначення загальних меж і контексту об'єкта моделювання на ранній стадії проєктування системи;

– сформулювати загальні вимоги до функціональної поведінки системи, що проєктується;

– розробити початкову концептуальну модель системи для подальшої розробки у вигляді логічної або фізичної моделі.

Діаграми варіантів використання - це діаграми, в яких вузли представляють акторів і вигоди, а зв'язки між вузлами представляють різні типи взаємовідносин;

Діаграми класів є центральними для проєктування об'єктно-орієнтованих систем. Діаграми класів визначають типи об'єктів і різні типи статичних зв'язків між ними. Діаграми класів мають властивість, спільну для всіх діаграм.

Це одна з проєкцій моделі. Що відрізняє цей тип діаграм від інших, так це те, що вони мають специфічний зміст. Діаграми класів зазвичай включають наступні сутності.

– класи;
– інтерфейси;
– взаємодії;
– залежності, узагальнення та асоціації.

Діаграми класів створюються для декількох систем, а не лише для однієї. Одна діаграма класів представляє підмножину класів, згрупованих в одному пакеті, і зв'язки між класами, а інша представляє ту саму підмножину разом з атрибутами і операціями класів. Діаграм класів створюється стільки, скільки потрібно для представлення системи.

Діаграма взаємодії представляє один з процесів обробки інформації варіанту використання; для одного потоку подій може бути створено декілька діаграм взаємодії. Основними елементами діаграми взаємодії є об'єкти.

Об'єкт — це сутність, яка включає в себе як дані, так і поведінку. Цей термін може використовуватись для позначення як реальних, конкретних об'єктів, так і абстрактних концепцій. Існує два основних види діаграм взаємодії: діаграми послідовностей і діаграми комунікацій. Діаграми послідовностей демонструють обмін повідомленнями між об'єктами вздовж часової осі, тоді як діаграми комунікацій відображають структуру взаємодії між ними. У діаграмах послідовностей зображається потік управління, а в діаграмах комунікацій — потік даних.

Зазвичай потік подій охоплює не одну лінійну послідовність дій, а кілька можливих варіантів розвитку подій. Кожен з варіантів може бути представлений певною послідовністю дій, або сценарієм. Таким чином, один випадок використання описує кілька можливих сценаріїв, кожен з яких відображає одну з ймовірних послідовностей дій.

Сценарій - це послідовність дій, що описує поведінку системи.

Діаграми послідовності дій показують взаємодію об'єктів у хронологічному порядку. «Обмеження» (constraints) не обмежують розширення елементів. Діаграма станів відображає еволюцію об'єкта від моменту його створення до знищення. Такі діаграми створюються не для всіх класів у системі, а лише для тих, що мають динамічну поведінку, активно обмінюються повідомленнями та змінюють свої стани. Хоча програмний код безпосередньо не генерується на основі діаграм станів, вони відіграють ключову роль у розумінні динаміки поведінки класу, що дозволяє краще усвідомити логіку змін перед початком кодування.

Діаграми розгортання представляють елементи та компоненти, що існують виключно під час виконання програми. На таких діаграмах відображаються лише екземпляри програмних компонентів, які є виконуваними файлами або динамічними бібліотеками. Компоненти, що не використовуються під час виконання, на діаграмі розгортання не показуються. Компоненти, що містять вихідний код, можуть бути представлені тільки на діаграмах компонентів і на діаграмі розгортання не відображаються.

Діаграми розгортання наочно демонструють процесори, пристрої, процеси та їхні зв'язки. Вони є єдиною діаграмою, яка відображає всю систему, оскільки включають специфікації реалізації. Зазвичай їх створення є фінальним етапом у процесі розробки моделі програмної системи. Основними елементами цієї діаграми є вузли та зв'язки між ними.

Метою створення діаграми розгортання є:

– визначити розподіл компонентів системи між фізичними вузлами;

– показати фізичні зв'язки між усіма вузлами реалізації системи під час виконання;

– виявити потенційні вузькі місця в системі та, при необхідності, реконфігурувати топологію для забезпечення оптимальної продуктивності;

– розробка починається на системному рівні і проходить через етапи аналізу, проєктування, кодування, тестування та супроводу [3].

Системний аналіз визначає роль кожного елемента комп'ютерної системи і те, як вони взаємодіють. Оскільки додаток є лише частиною великої системи, аналіз починається з визначення вимог до всіх елементів системи.

Кодування – це переклад результатів проєктування в текст будь-якою мовою програмування.

Тестування – це виконання програми з метою виявлення деяких дефектів у логіці, функціональності та формі програмного продукту.

Супровід вносить зміни до програмного забезпечення, щоб виправити помилки, відреагувати на зміни в середовищі або покращити програмне забезпечення.

Аналіз вимог охоплює елементи програмного забезпечення. На цьому етапі детально визначаються функції, можливості та інтерфейси програмного забезпечення. Всі рішення документуються в специфікації аналізу. На етапі проєктування створюються наступні типи:

- архітектура програмного забезпечення;
- модульна структура програмного забезпечення;
- алгоритмічна структура програмного забезпечення;
- алгоритмічна структура програмного забезпечення;
- структури даних;
- інтерфейси вводу-виводу.

При проєктуванні комп'ютерної системи необхідно розділити її на менші підсистеми.

Поширення персональних комп'ютерів, калькуляторів і поява потужних суперкомп'ютерів зробили комп'ютерне моделювання одним із найефективніших методів дослідження різних систем — фізичних, технологічних, біологічних, економічних та інших. Комп'ютерне моделювання часто виявляється простішим і зручнішим для дослідження складних явищ. За допомогою комп'ютера можна проводити експерименти, які важко реалізувати або передбачити їх результати.

Логічна та формалізована природа комп'ютерного моделювання допомагає виявляти основні фактори, що визначають характеристики досліджуваного об'єкта, та аналізувати реакцію системи на зміну початкових умов і параметрів. Процес комп'ютерного моделювання включає абстрагування від конкретних характеристик явища, створення спочатку якісної, а потім кількісної моделі. Наступний етап — проведення обчислювальних експериментів, інтерпретація їх результатів, порівняння отриманих даних із реальною поведінкою об'єкта та подальше вдосконалення моделі.

Суть комп'ютерного моделювання системи, етапи якої зображено на рисунку 1.1, полягає в описі поведінки елементів досліджуваної системи в процесі їх функціонування, розгляді їх взаємодії з навколишнім середовищем і проведенні серії комп'ютерних експериментів. Створюється комп'ютерна програма (пакет). Це робиться для вивчення властивостей і поведінки об'єктів, оптимізації їхньої структури та розвитку, а також для прогнозування нових явищ.

Рисунок 1.1 – Етапи комп’ютерного моделювання

Принципи комп'ютерного моделювання (рис 1.2). Зв'язок з іншими методами пізнання.

Модель – це об'єкт, який замикає досліджувану систему та імітує її структуру і створення. Модель може бути комп'ютерною програмою, поєднаною

з окремою моделлю фізичного об'єкта, системою математичних узагальнень або порівняльних даних.

Рисунок 1.2 – Схема комп’ютерного моделювання

Підпрограми імітують представлення основних характеристик об'єкта експертизи іншими комп'ютерними програмами в системі. Є наступні принципи моделювання:

– релевантність: модель, яка виділяє найважливіші аспекти, вивчає предмет і досліджує притаманні йому характеристики з прийнятною точністю. у цьому випадку результати моделювання можуть бути віднесені до предмета експертизи;

– простота та економічність: моделі повинні бути достатньо елементарними, щоб їх можна було використовувати ефективно та економічно доцільно. вони не повинні бути складнішими, ніж це необхідно для дослідження;

– достатність інформації: модель не може бути побудована без інформації про об'єкт моделювання. інформація повинна бути вичерпною у використовуваній моделі. існує таке поняття, як інформаційна достатність, яка дозволяє створити модель системи;

– реалістичність: створена модель повинна бути здатна достовірно завершити короткострокове дослідження;

– різноманітність та унікальність моделей: кожна модель характеризує лише певні аспекти реальної системи. для повного дослідження слід створити серію моделей, які відображають ключові аспекти існуючого процесу. кожна наступна модель повинна доповнювати та вдосконалювати попередню;

– принципи систематизації. парламентська система була задумана як серія взаємодій з іншими підсистемами і структурована за допомогою стандартних математичних методів. особливості системи не використовуються для її елементів;

– конфігураційні принципи. деякі з підсистем, що підлягають моделюванню, можуть бути визначені одним конкретним параметром (вектором, матрицею, графіком, формулою).

Моделі повинні відповідати наступним вимогам:

– бути лаконічною, тобто відображати найважливіші аспекти об'єкта з необхідною точністю;

– бути придатною для розв'язання конкретних типів задач;

– бути простою і зрозумілою, ґрунтуватися на мінімальній кількості описів і гіпотез;

– легко модифікуватися і доповнюватися;

– повинна бути простою у використанні.

Розрізняють наступні моделі:

– абстрактні моделі, тобто ментальні моделі, які існують лише в уяві. наприклад, структура алгоритму, яка може бути представлена блок-схемою, функціональними залежностями або диференціальними рівняннями, що

описують процес. До абстрактних моделей відносяться різні схематичні моделі, діаграми, структури та анімації;

– фізична модель (матеріальна модель) - це статична модель або пристрій управління, який працює певним чином. у реальному моделюванні створюються матеріальні моделі об'єктів і на них проводяться певні експерименти;

– абстрактні моделі поділяються на текстові, математичні та комп'ютерні. абстрактні моделі поділяються на текстові, математичні та комп'ютерні. текстові моделі включають в себе ряд речень на природній або формальній мові, що описують об'єкт пізнання.

Математичні моделі утворюють великий клас символічних моделей, які використовують математичні операції та оператори. Зазвичай це системи алгебраїчних або диференціальних рівнянь. Комп'ютерні моделі - це алгоритми або комп'ютерні програми, які розв'язують системи логічних, алгебраїчних і диференціальних рівнянь та імітують поведінку досліджуваної системи.

1.2 Аналіз засобів розробки

Для розробки автоматизованих систем моделювання поведінки людини в екстремальних ситуаціях необхідно проаналізувати програмні продукти, розроблені на мові UML. Прикладами є ArgoUML, StarUML, Visual Paradigm, Umbrello, MS Visio та UModel [4].

Для вибору програмного продукту для моделювання необхідно визначити критерії прийняття рішень. До таких критеріїв відносяться

– простота використання;
– доступність (відкрита ліцензія та повна функціональність);
– системні вимоги (низькі системні вимоги до ресурсів комп'ютера);
– імпорт та експорт з різних мов програмування;

– методології, що підтримують принципи об'єктно-орієнтованого моделювання.

Розглянемо детальніше програмні продукти для моделювання.

ArgoUML - програма для створення UML-діаграм, написана на мові Java і випущена під ліцензією Eclipse з відкритим вихідним кодом.

Основні можливості:

– підтримує специфікації uml 1.3 та 1.4. також підтримує дев'ять типів діаграм uml: координація, послідовність, діяльність, пріоритет, об'єкт, діаграма класів, стан, компонент і розгортання;

– генерує вихідний код на Java, C++ та PHP;

– підтримує зворотне проєктування з вихідного коду Java та байт коду;

– працює на платформах зі встановленою версією Java 5 або Java 6;

– надає можливість скасовувати/повторювати операції користувача;

– підтримує ocl;

– можливість збереження діаграм у форматах Gif, Postscript, eps та .svg;

– автоматична перевірка uml-моделей (критика дизайну);

Недоліки:

– не підтримує бази даних;

– немає інтеграції з іншими продуктами;

– не підтримує командну роботу;

– не підтримує створення звітів.

StarUML - це інструмент для моделювання програмного забезпечення з підтримкою UML; StarUML базується на UML версії 1.4 і підтримує підхід MDA (Model Driven Architecture) для реалізації концепцій профілів UML. StarUML базується на UML версії 1.4 і підтримує підхід MDA (Model Driven Architecture) до реалізації концепцій профілів UML. Середовище розробки добре адаптоване до потреб користувача і має високу розширюваність, особливо з точки зору функціональності. Воно добре адаптується до середовища розробки користувача

і може бути параметризоване для впливу на методи розробки програмного забезпечення, платформи проєктування і мови.

– основні можливості:
– суворо відповідає стандарту uml;
– відкритий формат для моделей програмного забезпечення;
– застосування методологій та платформ;
– можливості валідації програмних моделей;
– корисні плагіни для різних функцій.

Недоліки:

– відсутність готових об'єктів для прискорення роботи;
– не дуже зручний інтерфейс.

Visual Paradigm - це професійний UML-інструмент, який підтримує весь робочий процес розробки додатків: аналіз, проєктування, розробку та тестування. Він надає розробникам корисні посібники з UML, інтерактивні демонстрації UML та UML-проєкти.

Основні можливості:

– підтримує UML версії 2.1;
– підтримує генерацію коду на C#, VB.NET, Perl, Object-C та Ruby;
– підтримує зворотній інжиніринг для Object Definition Language, Flash ActionScript, Delphi, Java, C++, COBRA IDL, PHP, XML, Schema, Ada, Python, JDBC;
– підтримує EJB-діаграми та генерацію EJB-коду;
– інтегрується з Eclipse, NetBeans, InteliJ IDEA та Oracle JDeveloper;
– генерує звіти та документи у форматах PDF, HTML та Microsoft Word;
– операційне середовище - Linux, Mac OS та Windows;
– експорт графіків у форматах JPG, PNG, SVG, EMG та PDF.

Недоліки:

– ціни на ліцензії варіюються від 600 до 2000 доларів США.

Umbrello - безкоштовне програмне забезпечення для створення UML-діаграм на платформі. В основному використовується і розробляється у програмному середовищі KDE, а також гарно функціонує з іншими віконними менеджерами та іншими програмними середовищами.

– основні можливості:

– підтримує UML-діаграми всіх стандартних типів;

– генерує з діаграм вихідний код C++, Java, C#, PHP, JavaScript, ActionScript, SQL, Pascal, Ada, Python, IDL, XML Schema, Perl та Ruby;

– підтримує створення документації у форматах HTML та docbook;

– є вільно розповсюджуваним програмним продуктом;

– підтримує скасування/повторення операцій користувача.

– імпорт/експорт XMI файлів.

Недоліки:

– не підтримує бази даних;

– не інтегрується з іншими продуктами;

– режим роботи не використовується багатьма користувачами;

– операційне середовище - лише Linux та Unix.

MS Visio - частина офісного пакету Microsoft Office, призначена для створення різних типів діаграм.

Основні можливості:

– підтримує UML 2.0;

– підтримує бази даних MS Access, MS SQL Server, Oracle, IBM DB2, ODBC та OLE DB;

– режим роботи підтримує командну роботу. Існує інструмент перегляду для запису діаграм і нотаток, залишених іншими членами команди;

– доступні функції звітування;

– операційне середовище - Windows;

– операції користувача можуть бути скасовані/повторені;

– заздалегідь визначені фігури, перетягування та майстри у Visio Professional дозволяють створювати чіткі та інформативні діаграми;

Недоліки:

– не підтримує генерацію вихідного коду;

– не підтримує зворотне проєктування;

– ціни на ліцензії варіюються від $200 до $600.

UModel - це редактор UML, розроблений компанією Altova. Він пропонує широкий спектр можливостей, включаючи редагування зовнішнього вигляду та параметрів відображення елементів діаграм, а також керування деталями елементів на діаграмах.

Основні можливості:

– підтримує всі типи діаграм, включені в стандарт UML 2.1.1;

– генерує вихідний код для Java 1.4, Java 5.0, Java 6.0, C# 3.0, VB 7.1, VB8.0 та VB 9.0;

– зі зворотною інженерією для C#, VB.NET та Java;

– має плагіни для Microsoft Visual Studio та Eclipse;

– підтримує командну роботу;

– генерує документи у форматах HTML, Microsoft Word та RTF;

– операційне середовище - Windows;

– дії користувача можуть бути скасовані/повторені;

– реалізовано два додаткових типи діаграм: XML-діаграми та BPMN-діаграми.

Недоліки:

– не підтримує бази даних.

1.3 Висновки за розділом

Проаналізувавши вищезгадані програмні продукти для UML-моделювання, я прийшов до висновку, що всі вони мають всі необхідні функції для вирішення поставленої задачі. Тим не менш, я зупинив свій вибір на UML інструменті MS Visio. Він дозволяє легко створювати та налаштовувати діаграми, а також має велику кількість шаблонів. Я вже користувався цією програмою раніше і був знайомий з її простим інтерфейсом.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Визначення та аналіз вимог до системи

Вимога – це умова або можливість, якій повинна відповідати система, і Стандартний глосарій термінології програмної інженерії IEEE (1990) визначає це поняття більш детально. Вимога – це:

– умова або можливість, необхідна користувачеві для вирішення проблеми або досягнення мети;

– визначення умов або можливостей, які повинна мати система або компонент системи, щоб виконати контракт або відповідати стандарту, специфікації чи іншому офіційному документу.

Альтернативне визначення. Вимоги - це вхідні дані, які формують основу для проєктування та побудови комп'ютерної системи. Початкові дані можуть надходити з різних джерел і характеризуються недостатністю інформації, неоднозначністю та мінливістю. Перш за все, розробникам потрібні вимоги, щоб визначити часові та вартісні прогнози проєкту та узгодити їх із замовником. Тому значна частина вимог повинна бути зібрана і оброблена на ранніх стадіях

створення комп'ютерної системи. Типи вимог та як їх визначити вказано у таблиця 2.1.

На практиці, однак, процес збору, аналізу та обробки охоплює весь життєвий цикл системи.

Таблиця 2.1 - Інформація про типи вимог

Поняття	Визначення
Бізнес-вимоги	Багаторівнева бізнес- цілі організації або замовника системи
Бізнес-правило	Стандарт, який уможливлює або обмежує різні бізнес-процеси. Не вимагає програмного забезпечення, а виступає лише як джерело для конкретних вимог до програмного забезпечення.
Обмеження	Обмеження можливостей, доступних розробникам під час проєктування та розробки системи.
Зовнішні вимоги до інтерфейсу	Опис того, як програмне забезпечення взаємодіє з користувачами, іншими системами або програмними пристроями.
Характеристика	Одна або більше логічно пов'язаних функцій системи, які мають цінність для користувача і описані набором функціональних вимог.
Функціональні вимоги	Опис необхідної поведінки системи в визначених умовах
Нефункціональні вимоги	Опис характеристик і функцій, які повинна мати система і яким вона повинна відповідати.
Атрибут якості	Тип нефункціональної вимоги, що описує характеристики послуги або продуктивність продукту.
Системні вимоги	Вимоги до продукту, що складається з декількох підсистем, які можуть бути програмними або комбінацією програмного та апаратного забезпечення.

Вимоги до комп'ютерної системи можна розділити на три рівні: бізнес-вимоги, вимоги користувачів і функціональні вимоги. Кожна система має власний набір нефункціональних вимог.

Бізнес-вимоги описують, навіщо потрібна система, тобто цілі, яких організація має намір досягти за допомогою системи. Бізнес-вимоги описуються в документах про бачення та мандат.

Вимоги користувачів описують цілі та завдання, які користувачі мають вирішувати за допомогою продукту, створюючи таким чином цінність. Сфера користувацьких вимог також описує атрибути та характеристики продукту, які є важливими для задоволення потреб користувача. Одним із способів вираження цього типу вимог є варіанти використання. Вимоги користувача описують, що користувач повинен вміти робити з системою.

Функціональні вимоги визначають, як система повинна працювати за певних умов. Функціональні вимоги визначають, що розробники повинні створити, щоб користувачі могли виконувати завдання (вимоги користувачів) в рамках бізнес-вимог. Взаємозв'язок між цими трьома рівнями вимог має вирішальне значення для успіху проєкту. Специфікація вимог до програмного забезпечення також може бути звітом, створеним на основі інформації, що зберігається в інструменті управління вимогами.

Специфікації вимог до програмного забезпечення, взаємозв’язки між якими зображені на рисунку 2.1, використовуються при розробці, тестуванні, забезпеченні якості продукту, управлінні проєктами та інших пов'язаних з ними функціях.

Термін «системні вимоги» описує вимоги до ряду компонентів або підсистем (ISO/IES/IEEE 2011). У цьому сенсі «система» - це не просто інформаційна система. Коли ми говоримо про систему, ми маємо на увазі програмне забезпечення або підсистему програмного та апаратного забезпечення. Люди і процеси також є частиною системи, і функції, визначені в системі, можуть бути застосовані до людей. Дехто використовує термін «системні вимоги» для позначення детальних вимог до програмної системи.

Специфікація SRS містить нефункціональні вимоги на додаток до функціональних вимог.

До нефункціональних вимог відносяться Атрибути якості, також відомі як параметри якості або вимоги до рівня обслуговування. Вони описують різні варіації характеристик системи, такі як продуктивність і доступність, які важливі для користувачів, розробників і тих, хто обслуговує систему.

Інші нефункціональні вимоги описують зовнішній інтерфейс між системою та зовнішнім світом. До них відносяться з'єднання та комунікаційні інтерфейси з іншими програмними системами, апаратними пристроями та користувачами. Обмеження проєктування та реалізації накладають обмеження на вибір, який роблять розробники під час проєктування системи.

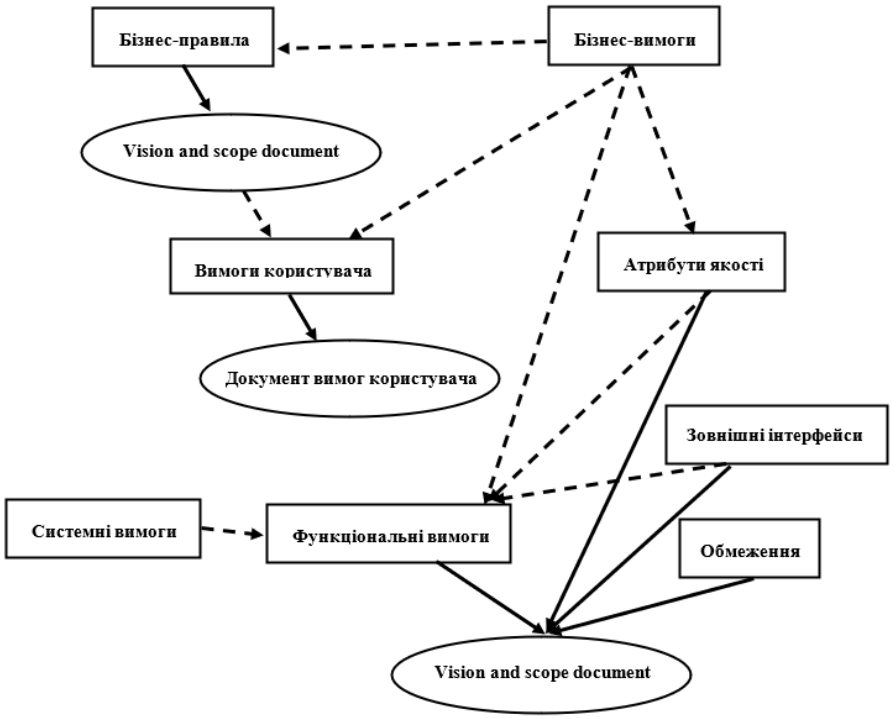

Рисунок 2.1 – Взаємозв’язки декількох типів інформації для вимог

Досі ми описували вимоги, які описують характеристики програмної системи, що створюється. Їх можна назвати вимогами до продукту. Однак вимоги до проєкту є принципово іншим типом вимог і відрізняються від них. Вимоги до проєкту включають наступне:

– фізичні ресурси, необхідні команді розробників, такі як робочі станції, спеціальні апаратні пристрої, тестові лабораторії та тестове обладнання;

– потреби в навчанні персоналу;

– документація для користувачів, така як навчальні матеріали, посібники, інструкції та інформація про випуски програмного забезпечення;

– документація з підтримки, наприклад, ресурси довідкової служби, інформація про технічне обслуговування та підтримку обладнання;

– зміни інфраструктури, які необхідно впровадити в операційному середовищі;

– вимоги та процедури для випуску, конфігурації, налаштування та тестування продуктів в операційному середовищі;

– вимоги та процедури міграції зі старої системи на нову, наприклад, вимоги до міграції та конвертації даних, налаштування безпеки;

– вимоги щодо відповідності сертифікації продукції та нормативним вимогам.

Сферу розробки технічних специфікацій, яка зображена на рисунку 2.2, слід розділити на розробку специфікацій вимог та управління вимогами. Розробка специфікації вимог може бути розділена на ідентифікацію вимог, аналіз, документування та затвердження.

Ідентифікація та збір вимог включає всі види діяльності, пов'язані з ідентифікацією вимог, такі як аналіз документів та створення прототипів. Основні види діяльності включають:

– визначення класів, очікуваних користувачами системи та іншими зацікавленими сторонами;

– розуміння завдань і цілей, а також бізнес-цілей, яким ці завдання відповідають;

– дослідити середовище, в якому буде використовуватися новий продукт;

– працювати з особами, що представляють кожен клас користувачів, щоб зрозуміти їхні потреби та очікування щодо якості.

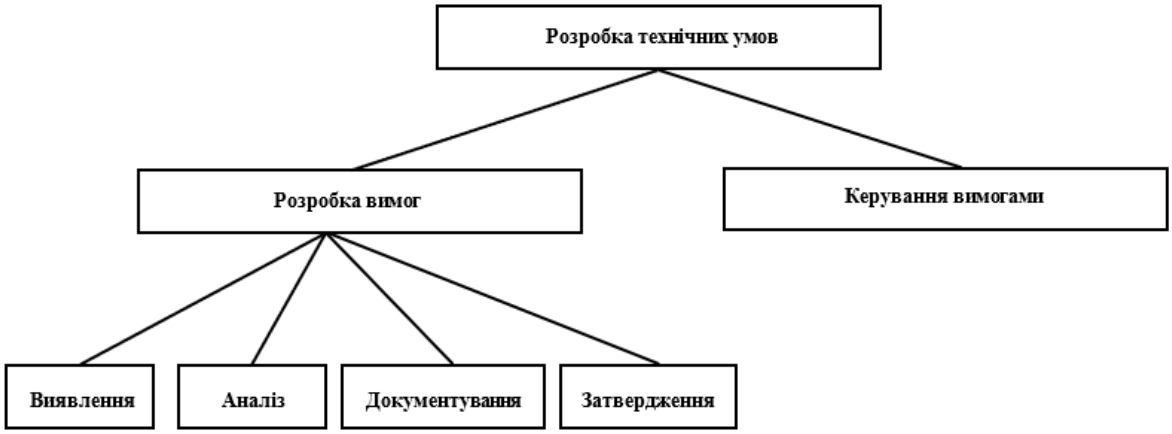

Рисунок 2.2 - Тематичні компоненти технічної специфікації

Аналіз вимог забезпечує більш широке і точне розуміння всіх вимог і виражає встановлені вимоги різними способами. Основні види діяльності є наступними:

– аналіз інформації, отриманої від користувачів, та відокремлення завдань користувачів від функціональних потреб, нефункціональних потреб, бізнес-правил, ймовірних рішень та іншої інформації;

– описати високорівневі вимоги до необхідного рівня деталізації;

– виводити функціональні вимоги з інформації про інші вимоги;

– розуміти відносну важливість атрибутів якості;

– призначати вимоги до програмних компонентів, визначених в архітектурі системи;

– погоджувати пріоритети реалізації;

– виявляти прогалини у вимогах або виявляти непотрібні вимоги, які не вписуються у визначені рамки.

Документувати вимоги: представляти та зберігати сукупність знань про вимоги в постійному та організованому вигляді.

Це включає переклад зібраних потреб користувачів у задокументовані вимоги та діаграми, придатні для розуміння, аналізу та використання цільовою аудиторією.

Валідація вимог передбачає перевірку правильності існуючого набору вимог, щоб розробники могли створювати рішення, які відповідають бізнес-цілям. Основні види діяльності включають:

– перевірка задокументованих вимог та усунення будь-яких недоліків до того, як вимоги будуть прийняті розробником;

– розробка приймальних тестів і стандартів, щоб гарантувати, що системи, створені на основі вимог, відповідають вимогам замовника і задовольняють бізнес-цілям.

2.2 Моделювання системи

Моделювання є фундаментальним методом дослідження в усіх галузях знань і науково обґрунтованим методом оцінки властивостей складних систем, що використовується для прийняття рішень. Модель - це опис об'єкта або суб'єкта, вихідної системи, яка замінюється іншою системою за певних умов або гіпотез з метою вивчення або відтворення її окремих характеристик. Моделювання - це процес побудови, вивчення та застосування моделей. Процес моделювання включає в себе абстрагування, формування гіпотез та конструювання. Моделі - це когнітивні інструменти, які дослідники ставлять між собою та об'єктом свого дослідження. Математичні моделі (аналіз та імітація) можуть бути використані для ефективного вивчення існуючих систем або систем, що знаходяться на стадії проєктування. Метою моделювання є стандартизація етапів розробки проєктів.

У процесі моделювання системи важливо визначити чітку і відносно просту внутрішню структуру. Моделювання повинно враховувати суперечливі вимоги.

Моделювання починається з формування понятійної системи, яка відображає характеристики об'єкта дослідження, тобто предметної області, що є необхідною для моделювання.

Найпоширенішим видом моделювання сьогодні є моделювання за допомогою різних інформаційних систем, побудованих на комп'ютерних моделях у вигляді прикладних програм різного призначення. Ці системи будуються з використанням сучасних об'єктно-орієнтованих методологій. Особливістю цієї методології є те, що в програмі описуються не процедури, необхідні для вирішення проблеми, а суб'єкти, які беруть участь у цих процедурах і забезпечують їх виконання. Це пов'язано з тим, що процедури виконуються не абстрактно, а через взаємодію декількох конкретних об'єктів, відповідно до тих властивостей, якими вони вже володіють. І все, що залишається після того, як ці об'єкти та їхні властивості описані в програмі, - це дати можливість цим об'єктам взаємодіяти. В результаті цієї взаємодії виконуються процедури, необхідні для вирішення поставленої задачі. Таким чином, об'єктно-орієнтоване моделювання здійснюється шляхом послідовного виконання:

Об'єктна модель, яка є концептуальною основою об'єктно-орієнтованих методів, має чотири основні елементи:

– абстрагування (виділення ознак, які відрізняють об'єкт від усіх інших типів об'єктів і, таким чином, чітке визначення характеристик цього об'єкта з точки зору подальшого аналізу);

– інкапсуляція (процес відокремлення елементів об'єкта. Інкапсуляція використовується для відокремлення контрактних зобов'язань щодо абстракції від її реалізації);

– модульність (характеристика системи, пов'язана з її потенційною можливістю бути декомпонованою на низку зовнішньо пов'язаних, але слабко взаємопов'язаних модулів);

– ієрархічність (абстракція впорядкованої системи).

Без будь-якого з цих елементів модель не є об'єктно-орієнтованою.

Абстрагування та інкапсуляція доповнюють одне одного. Абстракція призначена для відстеження поведінки об'єктів, в той час як інкапсуляція займається внутрішньою адаптацією. Іншими словами, клас складається з двох частин: інтерфейсу та реалізації. Інтерфейс відображає зовнішню поведінку, описуючи поведінку всіх об'єктів абстрагованого класу. Внутрішня реалізація описує представлення цієї абстракції та механізми досягнення бажаної поведінки об'єктів. Відокремлення інтерфейсу від реалізації дозволяє захистити об'єкт від деталей реалізації нижчого рівня [6]. Інкапсуляція дозволяє вносити зміни, зберігаючи надійність програми та мінімізуючи витрати на цей процес.

Об'єктно-орієнтований підхід реалізується за допомогою уніфікованої мови моделювання (UML).

Першим кроком у створенні моделі є створення діаграми варіантів використання (рис 2.3). Це найбільш загальне представлення функціонального призначення системи. Загальні діаграми практично не залежать від об'єкта моделювання і можуть бути використані в будь-якому проєкті, незалежно від об'єкта або області рішення. Варіант використання описує серію дій, які повинна виконати система у відповідь на подію, ініційовану зовнішнім об'єктом. На діаграмах варіантів використання використовуються два типи сутностей: варіанти використання та актори:

Дотримуючись кроків розв'язання задачі розробки моделі поведінки людини, для створення необхідної комп'ютерної системи необхідно виділити наступні структурні елементи:

– інтерфейс користувача для побудови початкових параметрів та середовище для тестування моделі;

– системний модуль для розрахунку поведінки людини на основі даних, отриманих з конструктора;

– модуль для візуалізації отриманих результатів;

– модуль для збору статичних даних з сеансів моделювання поведінки, аналізу даних та порівняння ефективності методів моделювання.

Пропонується розділити програмні елементи системи, що розробляється, на декілька модулів, які можна розширювати, кожен з яких має власний користувацький інтерфейс. Такий підхід дозволяє більш детально тестувати та перевіряти помилки на всіх рівнях моделювання поведінки людини в екстримальних ситуаціях.

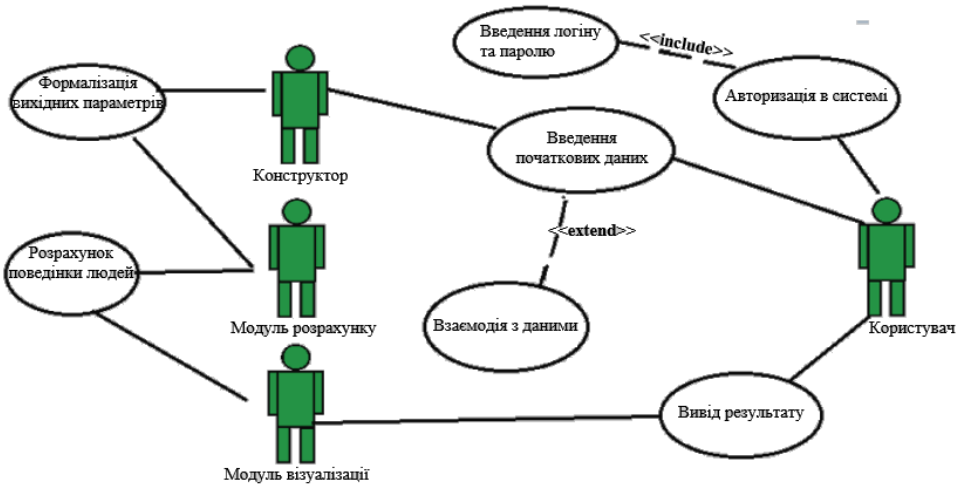

Рисунок 2.3 - Діаграма варіантів використання

Після запуску програми користувач повинен увійти в систему за допомогою імені користувача та пароля. Потім він може ввести початкові дані для моделювання поведінки. Вхідні дані включають кількість персоналу, площу приміщення, швидкість руху натовпу, ширину входів і виходів, кількість входів і виходів, щільність натовпу тощо. Потім розробник формалізує вхідні параметри, тобто приводить дані до певної форми. Потім розрахунковий модуль обчислює поведінку людей за певною формулою. Наступним кроком є імпорт даних до модуля візуалізації. Цей модуль аналізує дані та відображає їх

користувачеві. Кожен реалізований модуль є незалежним програмним продуктом, який взаємодіє з іншими модулями.

Діаграми послідовності показують лише об'єкти, які безпосередньо беруть участь у взаємодії, а не їхні статичні зв'язки з іншими об'єктами. Важливим елементом для діаграм послідовності є динаміка взаємодії об'єктів у часі. Діаграми послідовності візуально представляють послідовність процесу, дивлячись на обмін повідомленнями в часі.

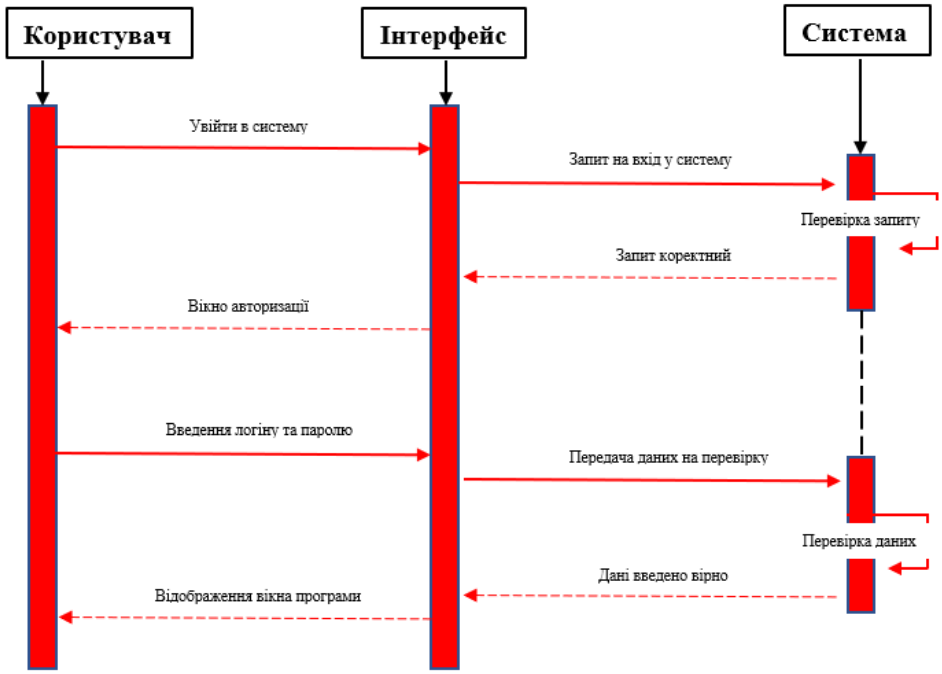

Рисунок 2.4 – Вхід у систему

Прецедент П1 - Вхід до системи.

Основний виконавець: користувач.

Необхідна умова: на комп'ютері запущено потрібний додаток.

Основна послідовність дій показана на рисунку 2.4.

– користувач натискає на кнопку входу в систему;

– інтерфейс надсилає запит системі на отримання дозволу для входу в додаток;

– система перевіряє отриманий запит на коректність;

– система надсилає відповідь інтерфейсу щодо коректності запиту;

– інтерфейс виводить користувачеві вікно авторизації;

– користувач вводить у вікні логін та пароль;

– інтерфейс надсилає отримані дані до системи на перевірку;

– система перевіряє правильність введених даних;

– система повертає інтерфейсу відповідь про успішне введення даних;

– система перевіряє коректність даних;

– система повертає інтерфейсу відповідь що дані були успішно введені;

– інтерфейс відображає вікно програми.

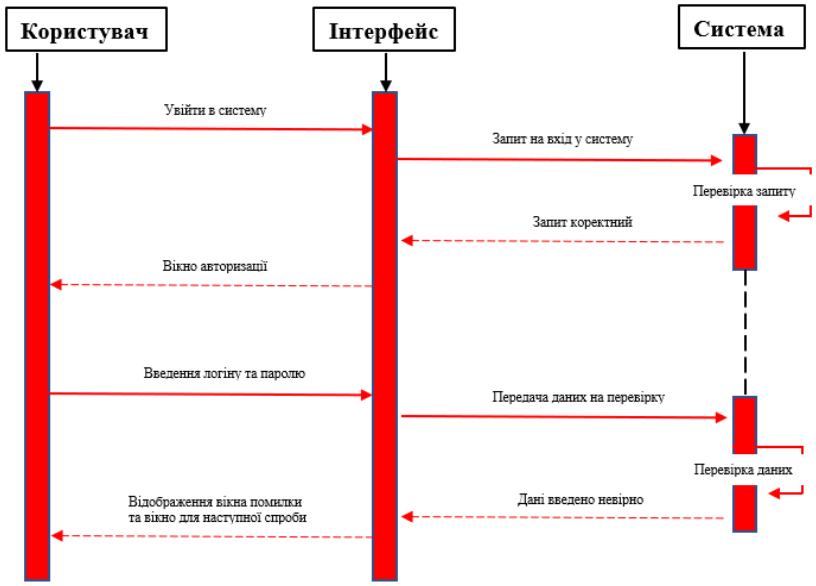

Рисунок 2.5 – Невдалий вхід в систему

Прецедент П2 – Невдалий вхід в систему.

Основний виконавець: користувач.

Необхідна умова: на комп'ютері запущено потрібний додаток.

Альтернативна послідовність збоїв показана на рисунку 2.5.

– користувач натискає на кнопку «Увійти»;

– інтерфейс запитує у системи дозвіл на вхід до додатку;

– система перевіряє коректність отриманого запиту;

– система повертає відповідь інтерфейсу щодо коректності запиту;

– система перевіряє коректність отриманого запиту;
– система повертає відповідь інтерфейсу;
– інтерфейс виводить користувачеві вікно авторизації;
– користувач вводить у вікні логін та пароль;
– інтерфейс надсилає отримані дані до системи на перевірку.
– система перевіряє правильність даних;
– система повертає інтерфейсу відповідь, що дані введені невірно;
– інтерфейс виводить вікно помилки і вікно для повторного входу.

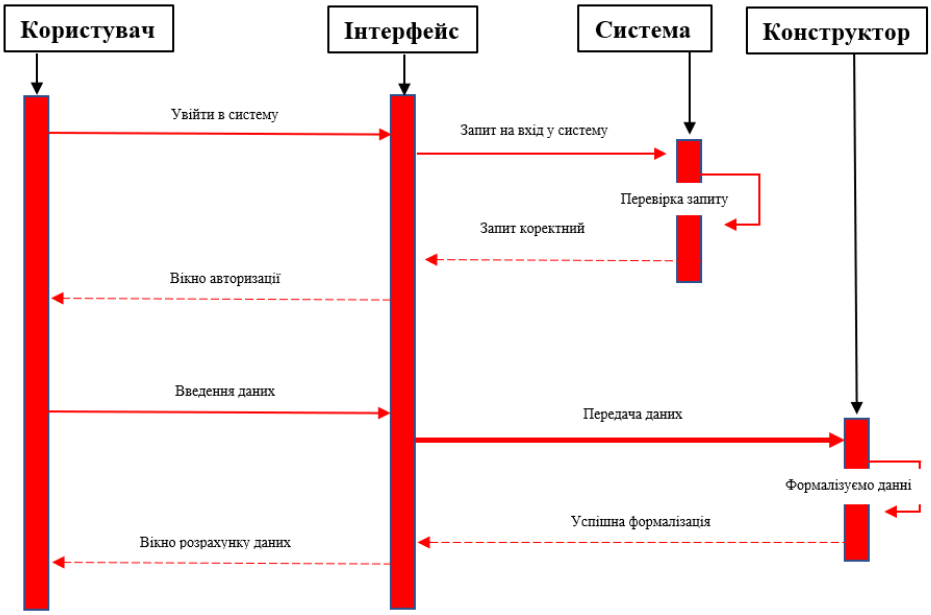

Рисунок 2.6 – Введення даних для моделювання

Прецедент П3 – Введення даних для моделювання.

Основний виконавець: користувач.

Необхідна умова: на комп'ютері запущено потрібний додаток.

Основний потік показано на рисунку 2.6.

– користувач натискає на кнопку «Ввести дані для моделювання»;
– інтерфейс надсилає запит у систему на отримання дозволу для введення даних;
– система перевіряє отриманий запит на коректність;

– система повертає інтерфейсу відповідь про коректність запиту;

– система надсилає інтерфейсу відповідь про коректність запиту;

– інтерфейс виводить користувачеві вікно введення даних;

– користувач вводить необхідні дані у форму;

– інтерфейс відправляє отриману інформацію до конструктора;

– конструктор форматує введені дані;

– конструктор повертає відповідь інтерфейсу про те, що дані успішно відформатовано;

– інтерфейс виводить вікно для обчислення даних.

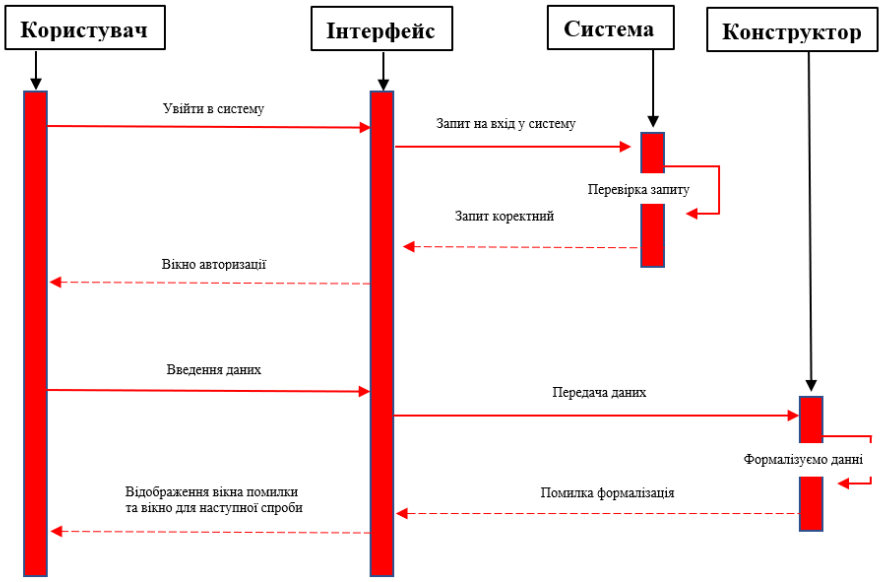

Рисунок 2.7 – Помилка введення даних для моделювання

Альтернативна послідовність – Рисунок 2.7.

– Користувач натискає на кнопку «Ввести дані для моделювання»;

– інтерфейс надсилає запит до системи на отримання дозволу на введення даних

– система перевіряє отриманий запит на коректність;

– система повертає інтерфейсу відповідь про коректність запиту;

– інтерфейс виводить користувачеві вікно введення даних ;

– користувач вводить необхідні дані у форму;

– інтерфейс передає отриману інформацію конструктору;

– конструктор форматує введені дані;

– конструктор повертає інтерфейсу повідомлення про помилку, повідомляючи, що не може відформатувати дані;

– інтерфейс виводить вікна для та для повторного введення даних.

Рисунок 2.8 - Розрахунок даних

Прецедент П4 – Розрахунок даних.

Основний виконавець: користувач.

Необхідна умова: Користувач ввів дані для моделювання.

Основний потік показано на рисунку 2.8.

– користувач натискає кнопку «Почати розрахунок»;

– інтерфейс передає отримані дані до розрахункового модуля;

– розрахунковий модуль зберігає та обробляє дані;

– розрахунковий модуль повертає інтерфейсу результати успішно завершеного розрахунку;

– інтерфейс виводить вікно з результатами розрахунку.

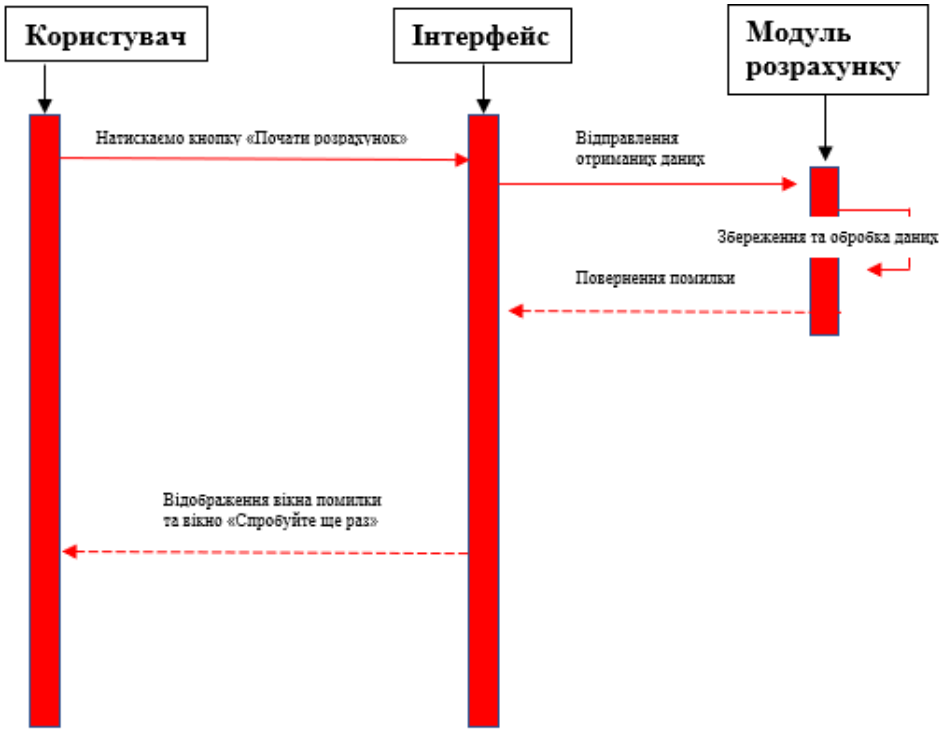

Рисунок 2.9 – Помилка при розрахунку даних

Альтернативна послідовність: Рисунок 2.9.

- користувач натискає кнопку «Почати розрахунок»;
- інтерфейс надсилає отримані дані до розрахункового модуля;
- розрахунковий модуль зберігає та обробляє дані;
- розрахунковий модуль повертає інтерфейсу помилку розрахунку даних;
- інтерфейс виводить опис помилки у вікні та пропонує користувачеві повторити розрахунок.

Прецедент П5 – Розрахунок даних.

Основний виконавець: користувач.

необхідна умова: Користувач розрахував дані для моделювання;

основний потік показано на рисунку 2.10;

- користувач натискає на кнопку «Розрахувати симуляцію»;
- інтерфейс надсилає отримані дані до модуля візуалізації;
- модуль візуалізації зберігає та обробляє отримані дані;
- модуль візуалізації повертає результати успішного моделювання;

– інтерфейс відображає результати моделювання у вікні.

Рисунок 2.10 – Візуалізація результату

Альтернативна послідовність – Рисунок 2.11.

– користувач натискає на кнопку «Розрахувати симуляцію»;

– інтерфейс надсилає отримані дані до модуля візуалізації;

– модуль візуалізації зберігає та обробляє отримані дані;

– модуль візуалізації повертає інтерфейсу помилку при моделюванні результату;

– інтерфейс виводить опис помилки та пропонує повторити спробу.

Рисунок 2.11- Помилка при візуалізації результату

Прецедент П6 – Отримання інформації про результати моделювання.

Основний виконавець: користувач.

Необхідна умова: На комп’ютері попередньо була запущена програма.

Основний потік показано на рисунку 2.12.

– користувач натискає на кнопку «Увійти»;
– інтерфейс надсилає запит до системи на дозвіл на вхід до додатку;
– система перевіряє отриманий запит на коректність;
– система повертає інтерфейсу відповідь про коректність запиту;
– система надсилає інтерфейсу відповідь про коректність запиту;
– інтерфейс виводить користувачеві вікно аутентифікації;
– користувач вводить у вікні логін і пароль;
– інтерфейс надсилає отримані дані до системи на перевірку;
– система перевіряє правильність введених даних;
– система повертає інтерфейсу відповідь про успішне введення даних;
– інтерфейс відображає вікно програми;
– користувач натискає на меню;

– користувач натискає на довідку;

– користувач натискає на довідку і вибирає «Інформація про моделювання»;

– інтерфейс відображає інформацію про моделювання у вікні.

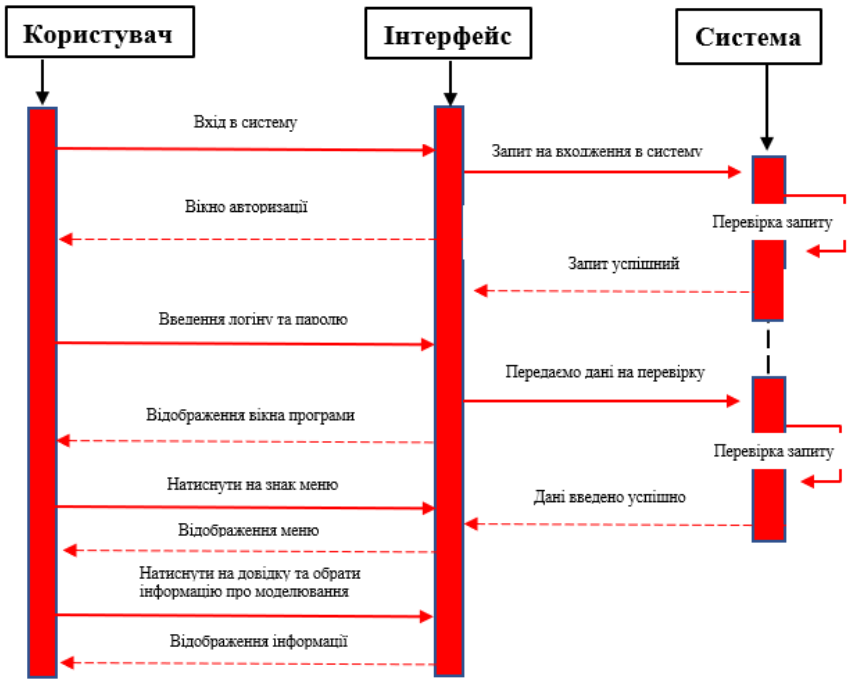

Рисунок 2.12 – Отримання інформації про результати моделювання

Важливою частиною об'єктно-орієнтованого проєктування є створення логічних моделей системи у вигляді діаграм класів (рис. 2.13). Діаграми класів використовуються для представлення статичної структури моделі системи у вигляді класів в об'єктно-орієнтованому проєктуванні. Діаграми класів показують різні взаємозв'язки між окремими сутностями в предметній області, такими як підсистеми та об'єкти, і можуть описувати їх внутрішню структуру і типи взаємозв'язків. Діаграми класів можуть включати інтерфейси, пакети, зв'язки та окремі екземпляри об'єктів і зв'язків.

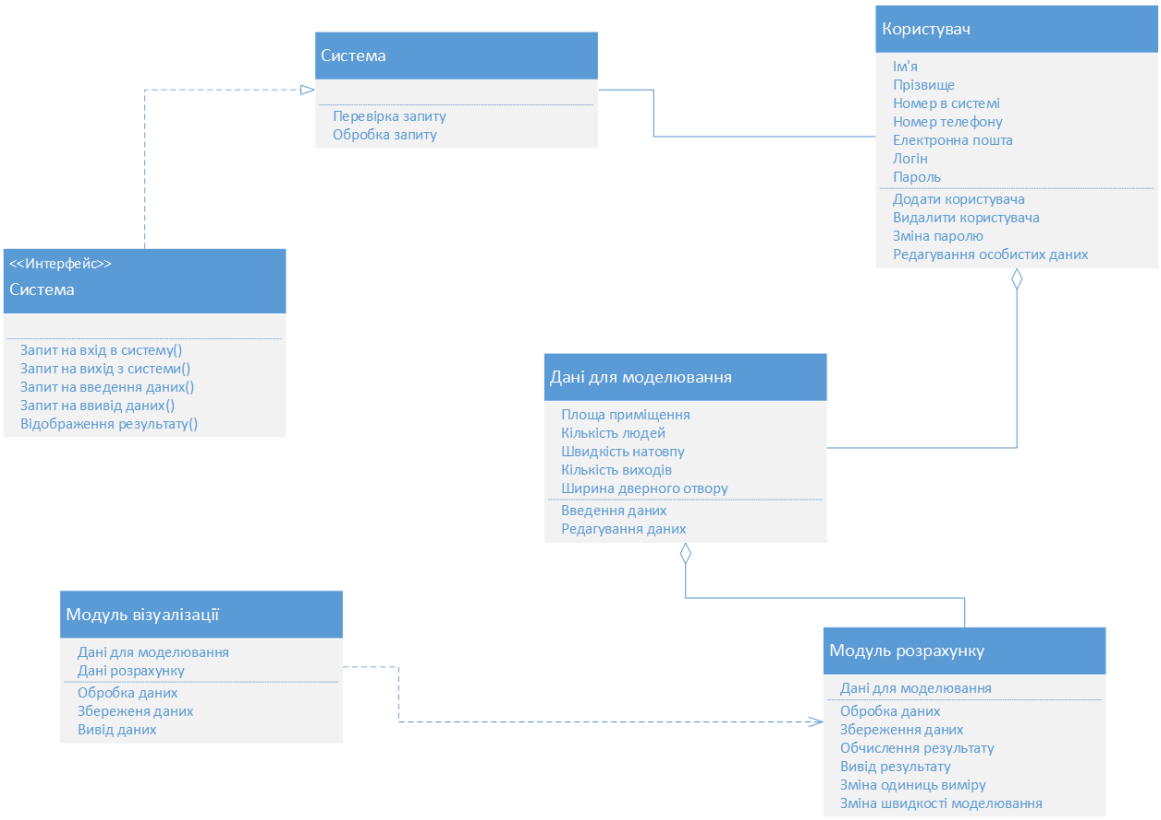

Рисунок 2.13 – Діаграма класів

Графічні класи представлені прямокутниками. Зазвичай вони поділяються на три сектори: ім'я класу, атрибути та операції.

Інтерфейсні класи можуть не мати атрибутів, як показано на схемі вище. Зв'язки між класами включають залежності, асоціації, узагальнення та реалізації.

Діаграма станів використовується для опису послідовності переходів об'єкта з одного стану в інший. Діаграма показує всі можливі стани, в яких може перебувати об'єкт, і процес, за допомогою якого його стан змінюється під впливом зовнішніх впливів. Загалом, діаграми станів підходять для опису однієї системи в межах декількох прецедентів [7].

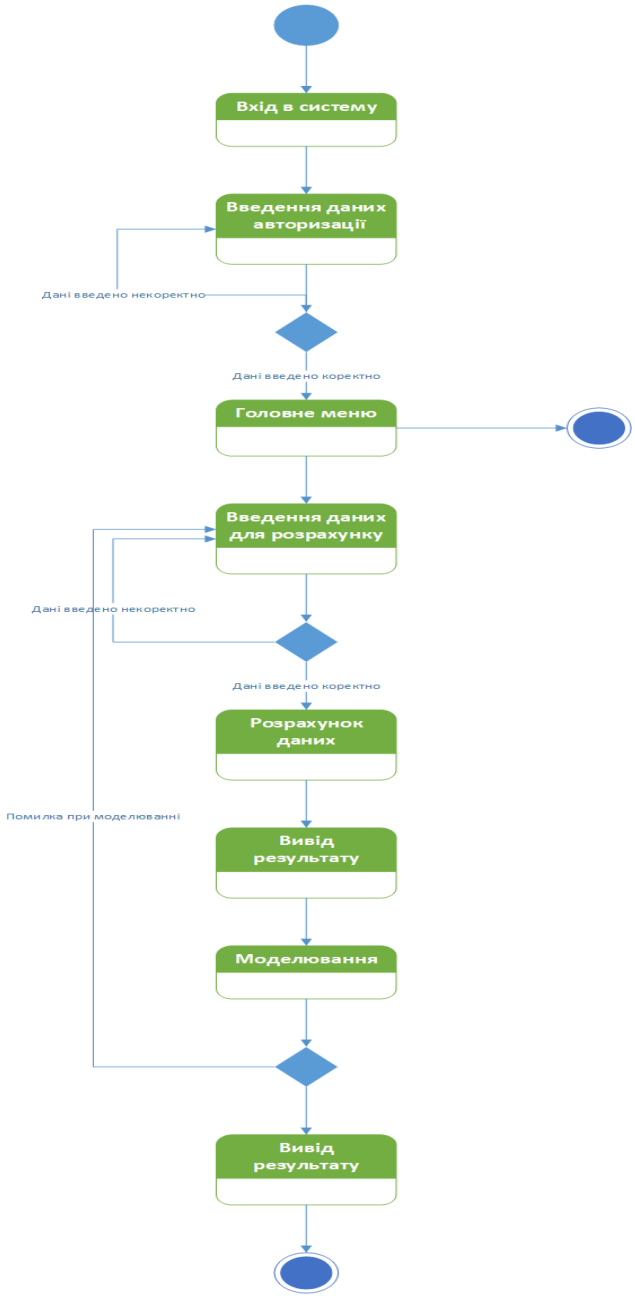

Рисунок 2.14- Діаграма стану

Однак діаграма вважається неефективною при формалізації поведінки декількох взаємодіючих об'єктів. Основними елементами діаграм є стани (рис. 2.14) та переходи. Властивості станів системи, слабо залежать від логічних структур, відображених на діаграмі класів. Тому стан системи може бути різним.

З цієї причини необхідно на деякий час вилучити їх зі структурних властивостей об'єктів і розглядати в зовсім іншій категорії, яка формує динамічний контекст поведінки модельованої системи.

Діаграми розгортання (рис. 2.15) використовуються для відображення елементів і компонентів програми, які можуть бути виявлені тільки під час виконання програми. У цьому випадку програма, будучи виконуваним файлом або динамічною бібліотекою, є лише екземпляром компонента. Компоненти, які не використовуються під час виконання, не відображаються на діаграмі розгортання. Тому компоненти з програмними джерелами можна знайти лише на діаграмах компонентів. Вони не відображаються на діаграмі розгортання.

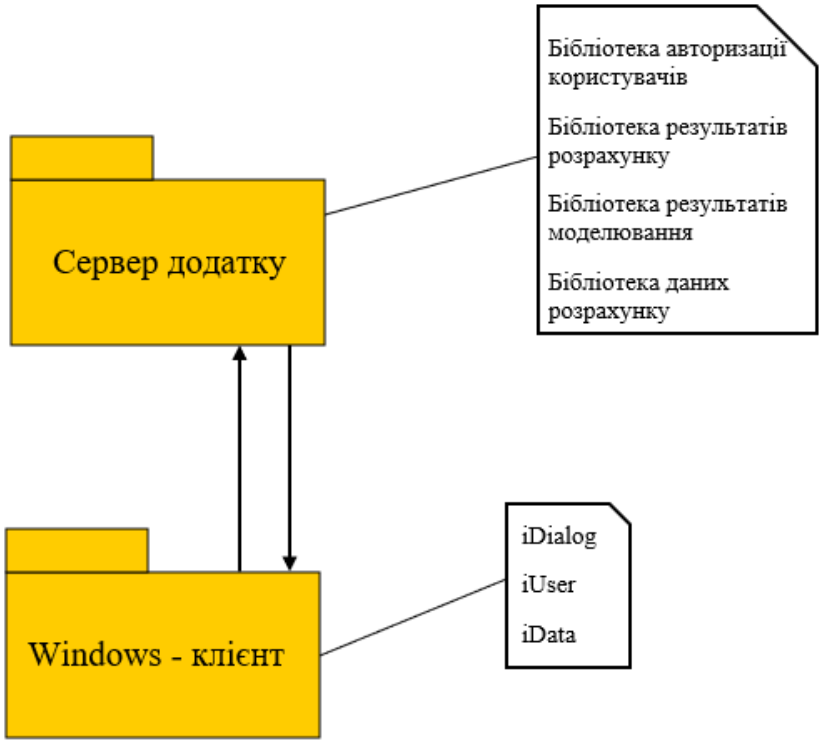

Рисунок 2.15 – Діаграма розгортання

Компонувальні схеми графічно представляють процесори, пристрої та процеси, а також їхні взаємозв'язки. На відміну від логічних схем, макетні схеми всієї системи є однорідними, оскільки вони повинні повністю відображати можливості реалізації. Фактично діаграма завершує процес створення конкретної програмної системи, розробка якої зазвичай є завершальним етапом проєктування моделі.

У системі, що моделює поведінку людини в екстримальній ситуації, компонент інтерфейсу iDialog розміщується на клієнті Windows, і передбачається, що користувачі можуть зареєструватися на сервері.

Бібліотека автентифікації користувачів може бути розміщена на сервері програми, так само як і бібліотека розрахункових даних. Інтерфейси iDialog повинні дозволяти користувачам вводити дані після автентифікації.

Бібліотека обчислюваних результатів розміщується на сервері додатків. Інтерфейс iUser використовується для того, щоб дозволити вже авторизованим користувачам отримати доступ до цієї бібліотеки з клієнта Windiws і отримати результати після того, як вхідні дані були обчислені. Інтерфейс iUser також може отримати доступ до бібліотеки результатів моделювання на сервері додатків для отримання результатів моделювання після отримання розрахованих даних і запуску моделювання. iData інтерфейс, Як на клієнті Windows, так і на сервері додатків можна відобразити бібліотеку результатів моделювання, в якій зберігаються результати розрахунків і попередні дані користувача.

Розширені алгоритми системи (рис. 2.16) наочно показують поведінку моделі і допомагають підтвердити загальний напрямок роботи моделі, приховуючи при цьому конкретні деталі реалізації і взаємодію окремих блоків схеми.

2.3 Проєктування інтерфейсу користувача

Графічний інтерфейс користувача (GUI) - це тип інтерфейсу користувача, елементи якого створені у вигляді графічних зображень [8]. Іншими словами, всі основні об'єкти, що відображаються в цьому інтерфейсі (символи, функціональні клавіші, об'єкти меню тощо), створені у вигляді зображень.

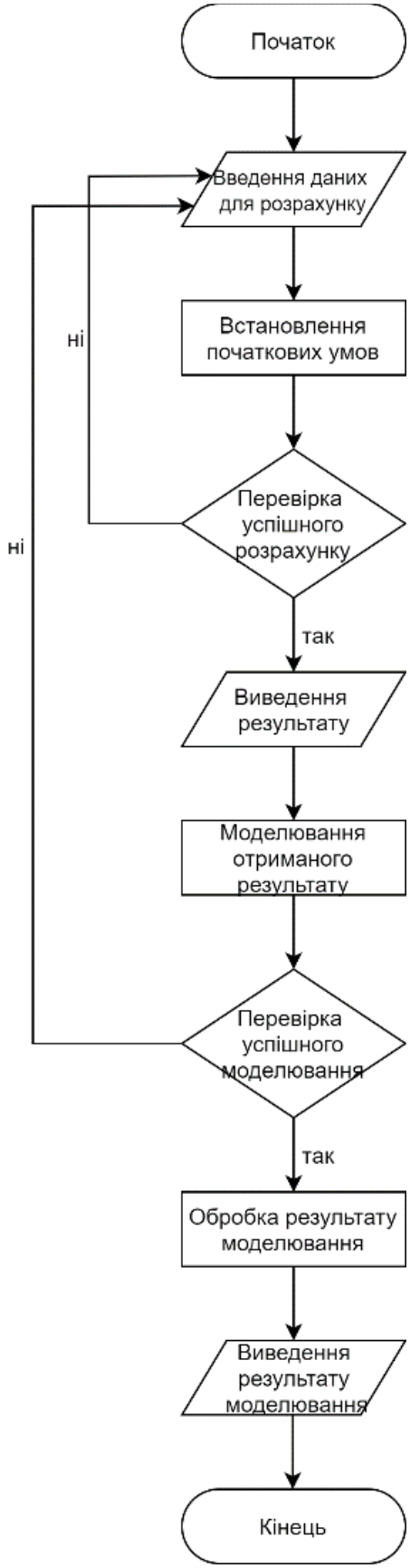

Рисунок 2.16 – Блок-схема алгоритму системи

Порівнюючи графічний інтерфейс користувача з традиційним командним рядком, користувачі першої версії мають повний доступ до всіх елементів, що відображаються на екрані. Цей доступ може бути досягнутий за допомогою різних пристроїв введення.

У графічному інтерфейсі кожен графічний об'єкт зазвичай передає значення функції за допомогою чіткого зображення. Це полегшує користувачеві маніпуляції з певним програмним забезпеченням або взаємодію з операційною системою в цілому. Однак важливо розуміти, що графічний інтерфейс - це лише частина графічного інтерфейсу користувача: графічний інтерфейс функціонує на рівні візуалізації даних і так само взаємодіє з користувачем.

Проєктування візуальної композиції та часової поведінки графічного інтерфейсу є важливою частиною прикладного програмування у сфері взаємодії людини з комп'ютером. Його мета - підвищити ефективність і зручність використання базової логічної конструкції програми. Методи проєктування, орієнтовані на користувача, використовуються для того, щоб візуальна мова, розгорнута в проєкті, працювала належним чином.

Інтерфейс користувача (UI) - це своєрідний канал зв'язку, через який взаємодіють користувачі та комп'ютери.

Найкращий користувацький інтерфейс - це той, який вимагає мало уваги користувача і є ледь помітним. Користувачеві не потрібно думати про те, які кнопки натискати або куди клацати мишею, він просто рухається. Ця поверхня повинна бути прозорою, щоб користувач міг бачити, над чим він працює.

Щоб створити ефективний користувальницький інтерфейс, який робить роботу з додатком приємною, потрібно знати, що користувач робить з додатком і які його вимоги до інтерфейсу. В цьому випадку ви є не тільки розробником, але і користувачем програми, оскільки ви можете побачити додаток очима користувача.

Дослідники HMI в Європі та США розробили основні принципи проєктування користувацьких інтерфейсів для комп'ютерних додатків. Як і в будь-якій науці, існують різні методи та класифікації.

Говорячи про найзагальніші принципи дизайну інтерфейсу користувача, можна виділити важливі моменти:

Перший принцип - це прозорість інтерфейсу, як уже згадувалося вище. Користувацькі інтерфейси повинні бути легкими для вивчення і не повинні бути перешкодою, яку користувач повинен подолати, щоб почати ними користуватися.

Другий принцип полягає в тому, щоб зосередитися лише на тому, що дійсно потрібно, а не на представленні очевидних елементів інтерфейсу. Цей принцип часто порушується авторами програмного забезпечення, які недооцінюють інтелектуальні здібності користувачів. Це було зрозуміло у 80-х і на початку 90-х років. Однак сьогодні такий підхід, який ігнорує користувача, явно недоречний. Робота з персональними комп'ютерами вимагає відносно невеликої початкової підготовки. Інтерфейс комп'ютерних програм, особливо операційної системи Windows, попередниці величезної індустрії програмного забезпечення, став простішим і доступнішим для людей.

Другим етапом розвитку графічного інтерфейсу став інтерфейс WIMP. Цей підтип інтерфейсу має наступні характеристики.

– вся робота з програмами, файлами та документами відбувається у вікні, тому певні частини екрану обмежені;

– всі програми, файли, документи, пристрої та інші об'єкти представлені піктограмами. Після відкриття піктограма стає вікном;

– всі дії над об'єктами виконуються за допомогою меню. Меню з'явилися на ранніх етапах створення графічних інтерфейсів користувача, але ніколи не відігравали домінуючої ролі і слугували лише доповненням до командного рядка. У чистому WIMP-інтерфейсі меню є основним елементом керування;

– маніпулятори широко використовуються для відображення об'єктів. Килимок для миші - це вже не просто іграшка, це вже не доповнення до клавіатури, а основний елемент маніпуляції. За допомогою маніпуляторів можна

вказувати на будь-яку область екрану, вікно або персонаж, виділяти їх і керувати ними за допомогою меню та інших прийомів.

Зверніть увагу, що для роботи з WIMP потрібен кольоровий екран з високою роздільною здатністю та вказівний пристрій. Крім того, програми, орієнтовані на цей тип інтерфейсу, мають вищі вимоги до обчислювальної потужності, пам'яті та пропускної здатності шини. З цих причин WIMP-інтерфейси наразі є найпоширенішим типом інтерфейсів.

Перше, що бачить користувач при вході в систему - це вікно авторизації (рис. 2.17).

– інформація для входу;
– інформація про пароль.

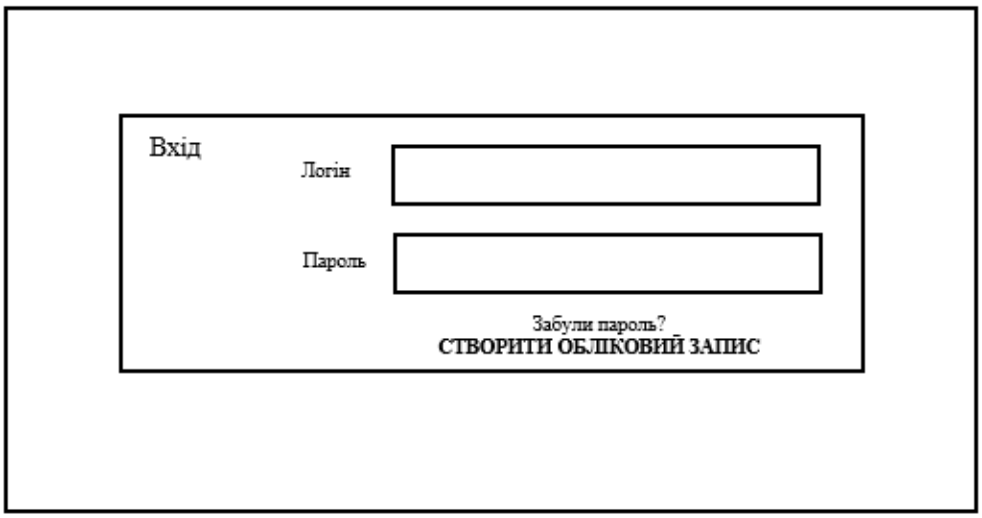

Рисунок 2.17 – Вікно авторизації

Після входу в систему користувач отримує доступ до головного меню (рис. 2.18). Спливаюче меню містить наступні пункти:

– налаштування;
– безпека;
– довідка
– вихід.

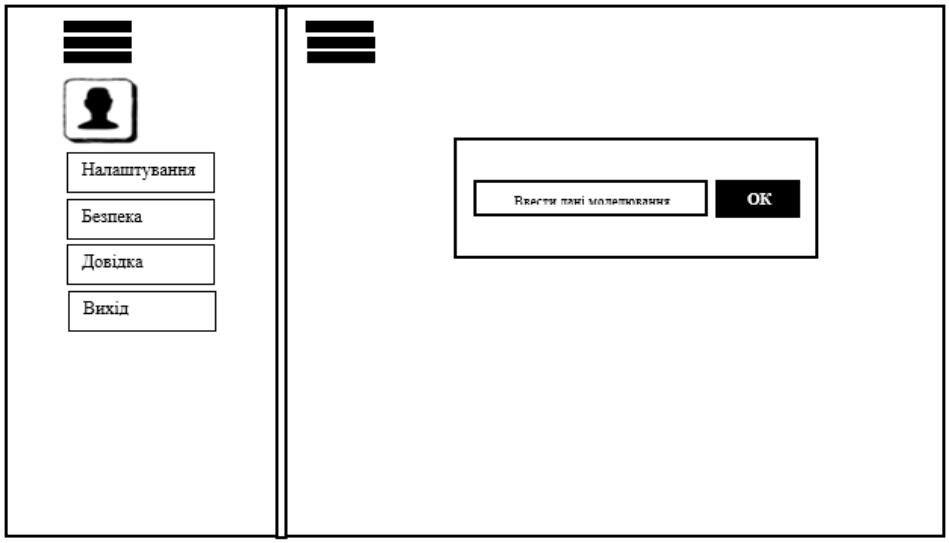

Рисунок 2.18 – Головне меню

При натисканні на кнопку «Введення даних для моделювання» відкривається вікно (рис. 2.19), що містить наступні елементи для отримання інформації для розрахунку:

– кількість людей;
– площа приміщення;
– щільність натовпу
– швидкість руху натовпу;
– ширина виходів;
– кількість виходів.

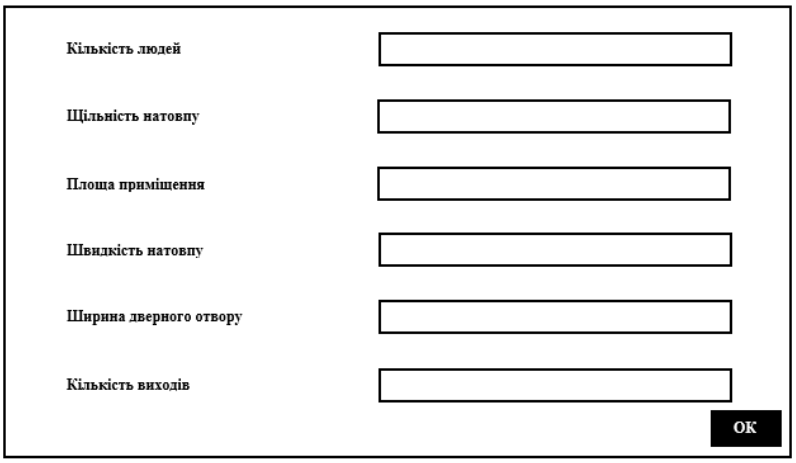

Рисунок 2.19 – Вікно вводу даних для моделювання

Коли введення даних і розрахунок пройшли успішно, з'явиться вікно результатів розрахунку (рис. 2.20).

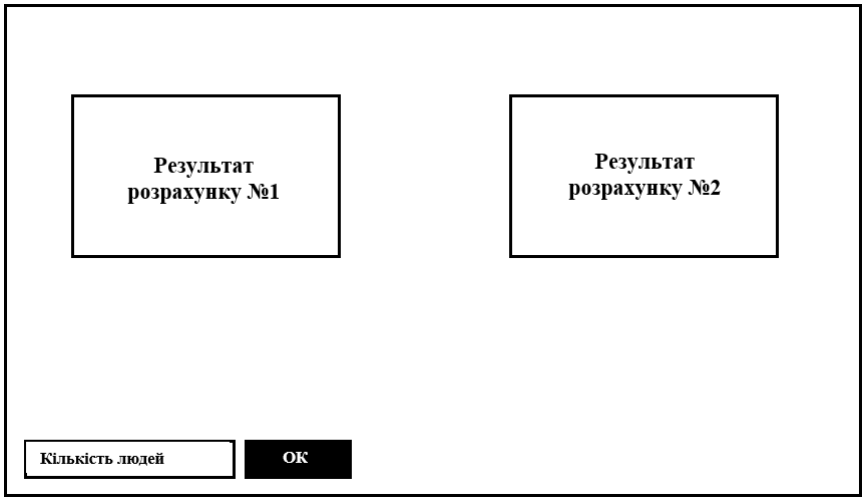

Рисунок 2.20 – Результат розрахунку

Останнє вікно - це результат моделювання (рис. 2.21), який відображається після натискання користувачем кнопки «Почати моделювання». Тут відображається візуалізація результатів розрахунків.

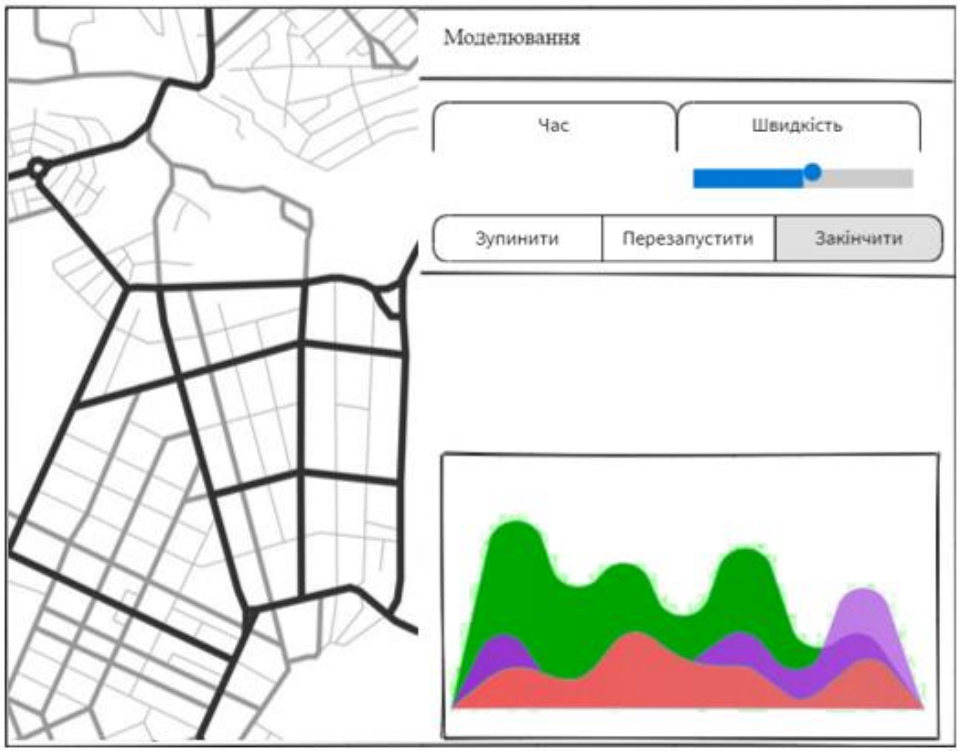

Рисунок 2.21 – Результат моделювання

3 РЕАЛІЗАЦІЯ СИСТЕМИ

Мова має багато різних застосувань, включаючи бекенд, веб-розробку, системне адміністрування, аналіз даних і машинне навчання. Python часто називають «мовою, що зв'язує».

Python має широкий спектр бібліотек, що надають готові до використання рішення для різних видів діяльності. Вони варіюються від базових математичних операцій до алгоритмів для складних завдань, таких як розпізнавання образів і розпізнавання мови.

Сьогодні Python стає все більш популярною серед програмістів і є вступною мовою програмування в університетах США. Згідно з опитуванням Інституту інженерів з електротехніки та електроніки (IEEE), Python входить до четвірки найпопулярніших мов програмування в США разом з Java, C та C++.

Програми на Python часто працюють повільніше, ніж програми, написані на Java. Однак, з іншого боку, час, необхідний для розробки на Python, значно скоротився, і програми на Python зазвичай у кілька разів коротші, ніж аналогічні програми на Java. Ця різниця може бути пов'язана зі спільними типами даних, що використовуються в Python. Наприклад, при програмуванні на Python не потрібно витрачати час на визначення типів змінних та аргументів.

Синтаксис Python простий. Це дозволяє легко читати і писати програмний код. Це прискорює розробку додатків і полегшує роботу програміста. Крім того, читабельний код легко підтримувати та модифікувати.

На відміну від Javascript, Python має добре реалізований об'єктно-орієнтований підхід, з класами та успадкуванням, які відіграють важливу роль у написанні великих програм і покращують повторне використання коду.

Python можна інтегрувати з іншими мовами, що використовуються на підприємстві, такими як Java та .Net; Python також може безпосередньо викликати код на C та C++.

Python спочатку був розроблений для інтеграції. Тому він підходить для адаптації великих додатків і створення розширень.

Tkinter - це пакет Python, призначений для роботи з бібліотекою Tk, яка містить компоненти графічного інтерфейсу користувача (GUI), написані мовою програмування Tcl [9].

Існує ряд бібліотек графічного інтерфейсу користувача, і хоча Tk навряд чи є найпопулярнішою, багато проєктів було створено з використанням Tk. Однак для Python її було обрано за замовчуванням з кількох причин: інсталяційний файл Python зазвичай включає пакунок tkinter як бібліотеку за замовчуванням, разом з іншими модулями.

Tkinter можна описати як перекладач з Python на Tcl: коли ви пишете програму на Python, код у модулі tkinter, що стоїть за нею, перекладає інструкції на Tcl, який бібліотека Tcl може зрозуміти.

Лістинг 3.1 – Вікно входу в ситему

```
from tkinter import *

root =Tk()
root.title('Система моделювання')
root.geometry('1200x720')
canvas = Canvas(root, width=1200, height=720, bg='#404040')
but1 = Button(root,
text= "Увійти в систему",
width=30,height=5,
bg="#F2F2F2",fg="#FF7F50")
but1.place(relx=.5, rely=.5, anchor="c", height=30,
width=150, bordermode=OUTSIDE)
canvas.pack()

root.mainloop()
```

Модуль Pickle надає функції та класи для серіалізації та десеріалізації об'єктів. У програмуванні серіалізація - це перетворення всіх даних у набір байтів, які зазвичай зберігаються у файлі або передаються мережею. Десеріалізація - це відновлення об'єкта з його байтового представлення.

Серіалізація часто використовується для запису даних користувача між різними сеансами програми (зазвичай гри). Простіший приклад: ви переглядаєте веб-сторінки і створили список або словник, який ви можете використати наступного разу, коли будете в мережі; скористайтеся функцією dump() у модулі Pickle, щоб зберегти об'єкт у файлі і вивантажити його наступного разу, коли будете завантажувати його за допомогою load().

Лістинг 3.2 – Функція реєстрації

```
def registration():
                text = Label(text="СТВОРИТИ ОБЛІКОВИЙ ЗАПИС")
    text_log = Label(text="Введіть логін:")
    registr_login = Entry()
    text_password1 = Label(text = "Введіть пароль:")
    registr_password1 = Entry()
text_password2 = Label(text = "Введіть пароль ще раз:")
    registr_password2 = Entry(show="*")
    button_registr =
Button(text="Зареєструватись",command=lambda:save())
    text.pack()
    text_log.pack()
    registr_login.pack()
   text_password1.pack()
    registr_password1.pack()
text_password2.pack()
    registr_password2.pack()
    button_registr.pack()
                def save():
        login_pass_save = {}
        login_pass_save[registr_login.get()] =
registr_password1.get()
        f = open("login.txt","wb")
        pickle.dump(login_pass_save, f)
        f.close()
        login()
    def login():
    text_log = Label(text="Вітаю!Теперь Вы можете війти в
систему")
    text_enter_login= Label(text="Введите логін:")
    enter_login = Entry()
    text_enter_pass = Label(text="Введітьпароль:")
    enter_password= Entry(show="*")
    button_enter=Button(text="Вхід",command = lambda:

registration()
root.mainloop()
```

Лістинг 3.3 – Функція входу

```
log_pass())
    text_log.pack()
    text_enter_login.pack()
    enter_login.pack()
    text_enter_pass.pack()
    enter_password.pack()
    button_enter.pack()
                def log_pass():
        f = open("login.txt","rb")
        d = pickle.load(f)
        f.close()
        if enter_login.get() in d:
            if enter_password.get() == a[enter_login.get()]:
                messagebox.showinfo("Ви війшли в
систему","Вітаю в системі")
            else:
  messagebox.showerror("Помилка!","Логін або пароль введені
невірно")
        else:
            messagebox.showerror("Помилка!","Логін невірний")
```

Модуль mpi4py копіює задану програму на одне з вказаних користувачем ядер процесора та інтегрує результати після використання методу build().

Лістинг 3.4 – Модуль для моделювання

```
from mpi4py import MPI

speed = MPI.COMM_WORLD.Get_speed()
size = MPI.COMM_WORLD.Get_size()
name = MPI.Get_processor_name()
```

Модуль для розрахунків складається з трьох головних функцій:

– producer() – збір даних;

– server() – обчислення даних;

– consumer() – отримання результатів.

Лістинг 3.5 – Модуль розрахунку

```
def server(input_q,next_q,i):
    while True:
        item = input_q.get()
        if i==0:item.st=time.time()
        timc.sleep(random.expovariate(glambda|i]))
```

```
         if i==M-1 :item.st=time.time()-item.st
            next_q.put(item)
            input_q.task_done()
        print("Server%d stop" % i)

def producer(sequence,output_q):
    for item in sequence:
        time.sleep(random.expovariate(glambda[0]))
        output_q.put(ilem)
def consumer(input_q):
    "Завершення процесу"
    ptime=time.time()
    in_seq=[]
    while True:
        item = input_q.get()
        in_scq+=[item]
        input_q.task_done()
        if item.cid == N-1:
            break
    print_results(in_seq)
    print("Кінець")
    print("Processing time sec. %d" %(time.time()-ptime))
```

4 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Тестування клієнтської частини додатку

Ручне тестування є частиною контролю якості тестування процесорів і імітується та виконується тестувальниками або звичайними користувачами.

Завдання тестувальника - знайти більшість помилок. Найпоширеніші помилки повинні бути ідентифіковані і знайдені в найкоротші терміни. Решту нетипових помилок можна розпізнати лише за допомогою ретельно розроблених інструментів.

Коли додаток використовується, очікується, що він буде виконувати завдання, для яких він був розроблений. Функціональне тестування проводиться для того, щоб переконатися, що кінцевий продукт відповідає технічним специфікаціям, що всі функції працюють належним чином і що додаток не містить помилок.

Функціональне тестування, як і будь-яке інше тестування, проводиться для того, щоб зробити продукт простим у використанні і уникнути помилок. Процес тестування повинен бути максимально наближений до нормального використання програми. Для забезпечення максимальної надійності аналізуються часто використовувані ресурси і те, як користувачі їх використовують.

Все програмне забезпечення реалізується в першу чергу для користувача. Тому всі функції повинні бути функціональними і простими у використанні. Найпоширеніші програмні компоненти встановлені в панелі управління.

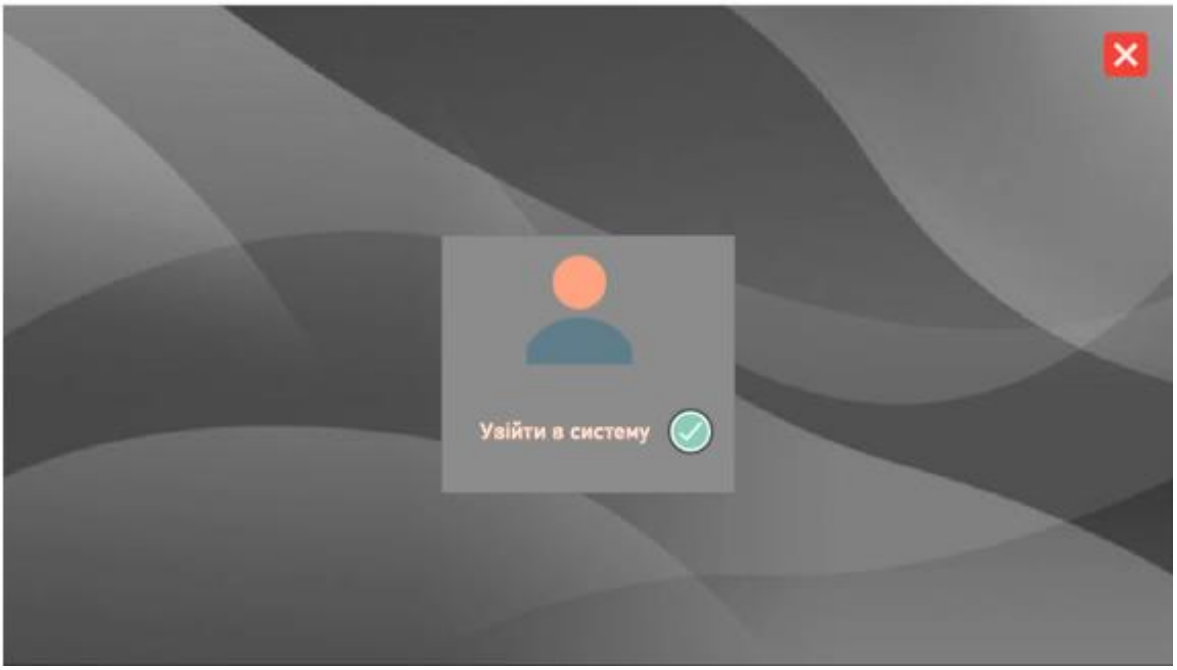

Рисунок 4.1 – Вхід в систему

На рисунку 4.1 показано вікно входу, яке з'являється, коли користувач запускає додаток. Користувач натискає на кнопку, позначену галочкою. Після цього з'являється вікно автентифікації системи.

У вікні автентифікації користувач повинен ввести правильне ім'я користувача та пароль. Якщо все введено правильно, користувачеві надається доступ до системи. Якщо введено неправильно (рис. 4.2), програма інформує

користувача про помилку у вигляді спливаючого вікна і повертає у вікно автентифікації.

Якщо користувач не зареєстрований в системі, необхідно натиснути на кнопку «Створити обліковий запис» (рис. 4.3). Після цього з'явиться вікно з формою для введення номера телефону, імені, прізвища, адреси електронної пошти та логіна/пароля. Система збереже дані та надішле на вказану електронну адресу лист із підтвердженням. Після цього ви можете увійти в систему.

Якщо ви забули свій пароль, ви можете його відновити. Для цього натисніть на кнопку Забули пароль у вікні входу в систему (рис. 4.4).

Рисунок 4.2 – Дані введено неправильно

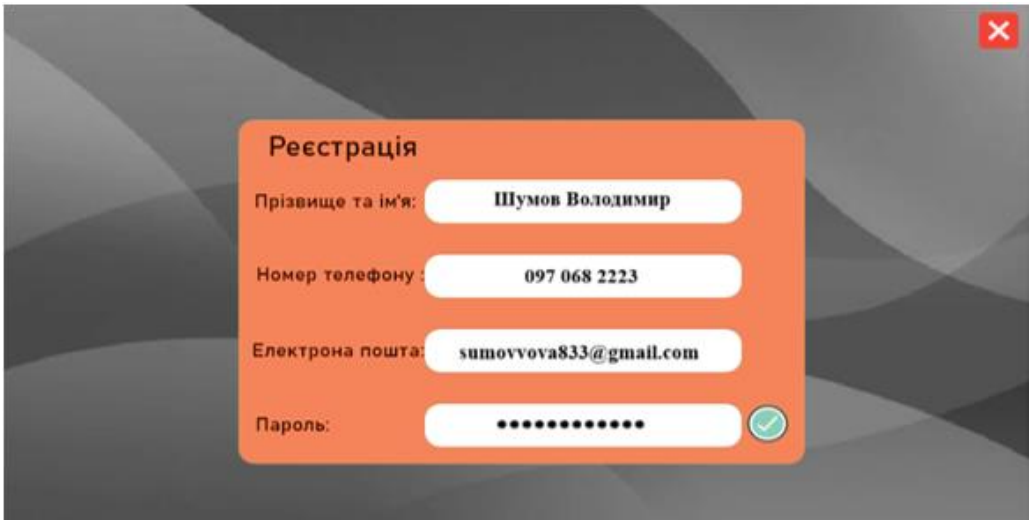

Рисунок 4.3 – Реєстрація

З'явиться вікно, де ви можете ввести свою електронну адресу, і на неї буде надіслано лист з паролем. Після цього ви можете спробувати увійти ще раз. Після успішної автентифікації з'явиться головна сторінка з меню у верхньому лівому куті (Рис. 4.5). Головне меню містить наступні кнопки:

– кнопка «Налаштування» використовується для зміни мови інтерфейсу та налаштування параметрів моделювання (зміна одиниць вимірювання та швидкості моделювання);

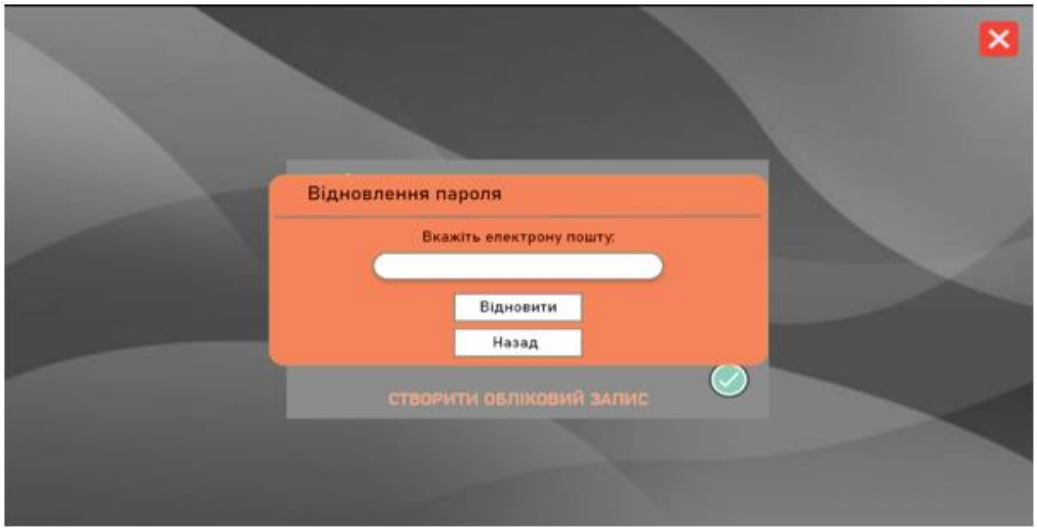

Рисунок 4.4 – Відновлення пароля

– кнопка «Безпека» - для заміни паролю, видалення облікового запису, видалення збережених даних після завершення моделювання;

– кнопка «Допомога» - для отримання інформації про попередні результати моделювання;

– кнопка «Вихід» - для виходу з системи.

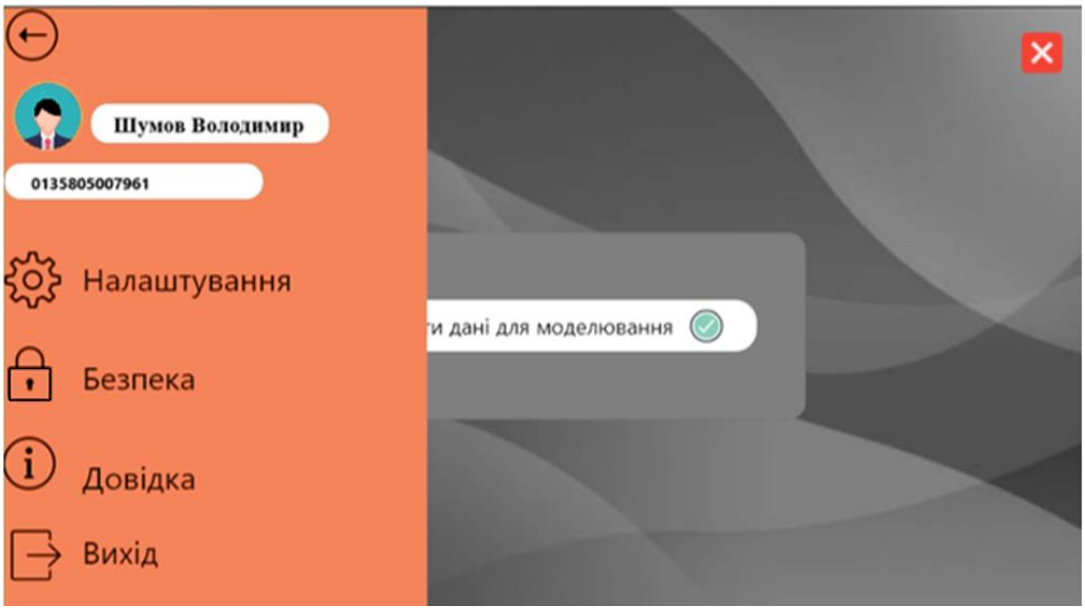

Рисунок 4.5 – Домашня сторінка

Окрім меню, на головній сторінці є кнопка «Введення даних для моделювання». Коли користувач натискає на неї, система відображає вікно з формою введення даних (рис. 4.6), і користувач вводить значення, що відображаються на екрані (кількість людей, щільність натовпу, площа приміщення, швидкість натовпу, ширина входів і виходів).

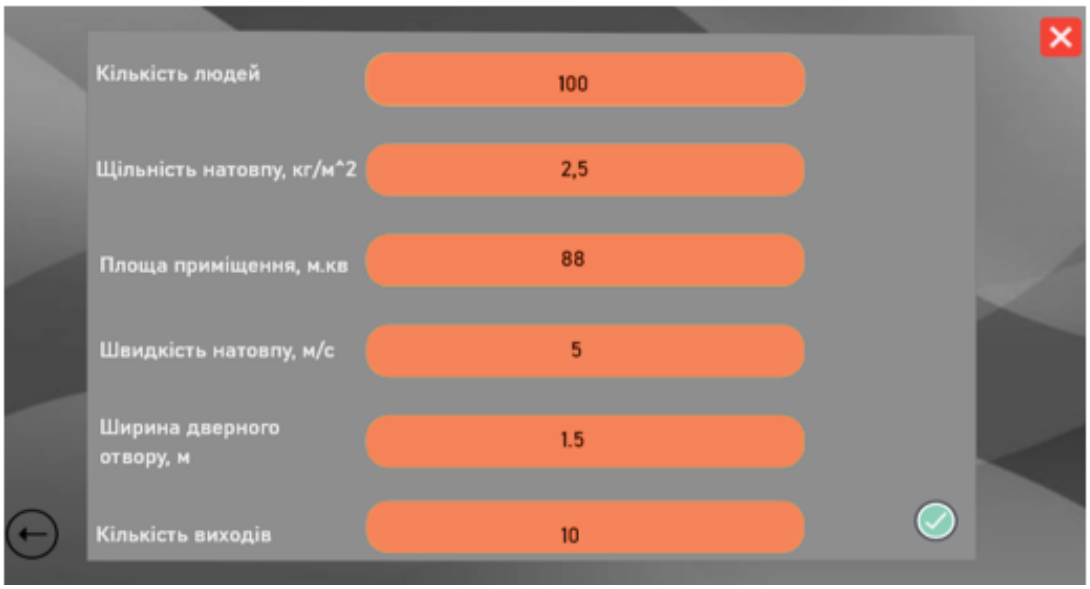

Рисунок 4.6 –Введення даних для моделювання

Після введення значень користувач натискає на кнопку підтвердження. З'являється наступне вікно (рис. 4.7), і користувач може або почати розрахунок даних, або повернутися до вікна введення даних для редагування.

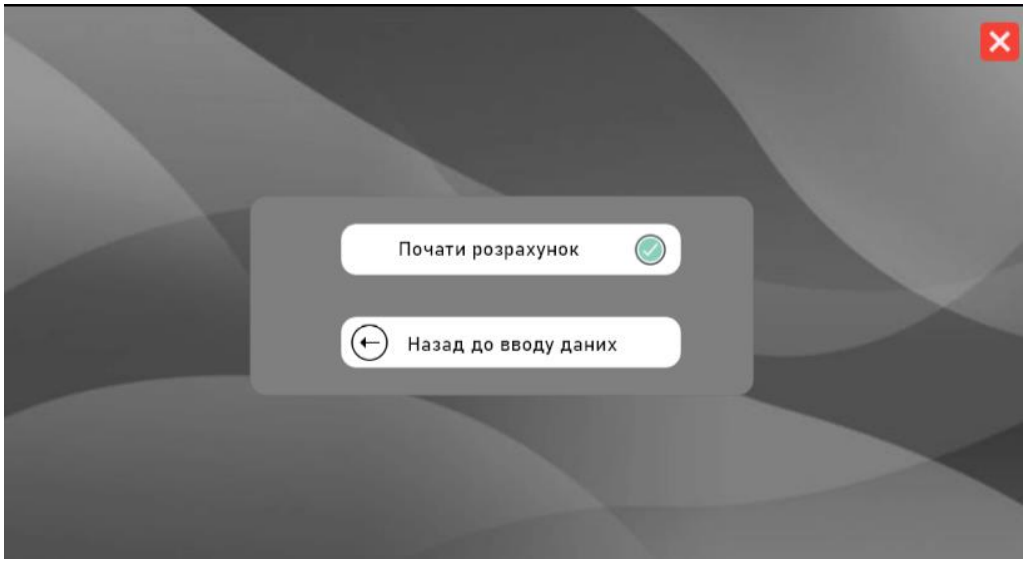

Рисунок 4.7 – Розрахунок даних

Якщо користувач натискає кнопку «Почати розрахунок» і розрахунок проходить успішно, програма повідомляє про це користувача, відображаючи спливаюче вікно з результатами розрахунку (рис. 4.8).

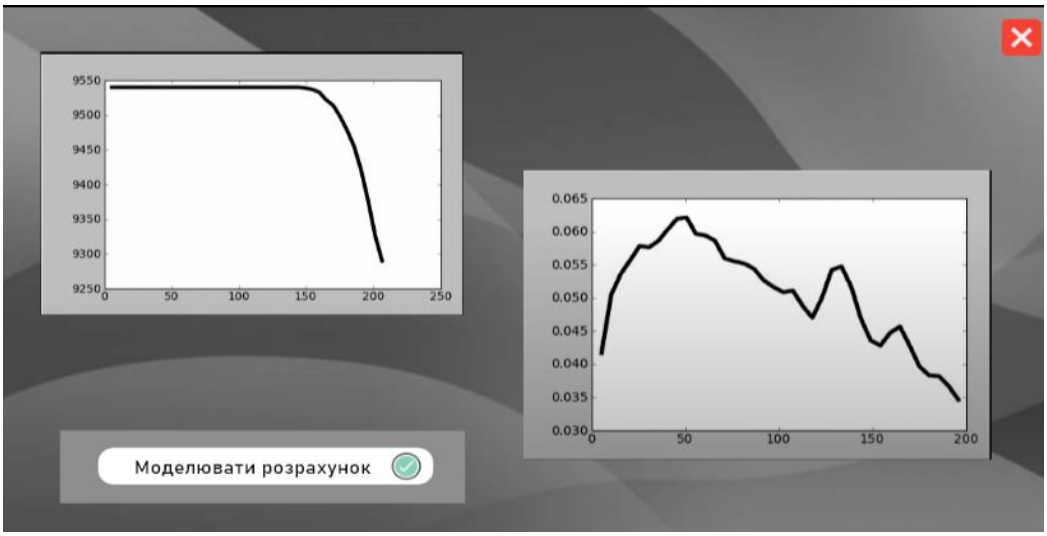

Рисунок 4.8 – Виведення результату розрахунку

Якщо користувач помиляється і вводить дані неправильно, система сповіщає його про це і відображає вікно, в якому вказується, що сталася помилка в розрахунках (рис. 4.9), і користувачеві пропонується спробувати ввести дані ще раз. Система повертає користувача до вікна введення даних для моделювання.

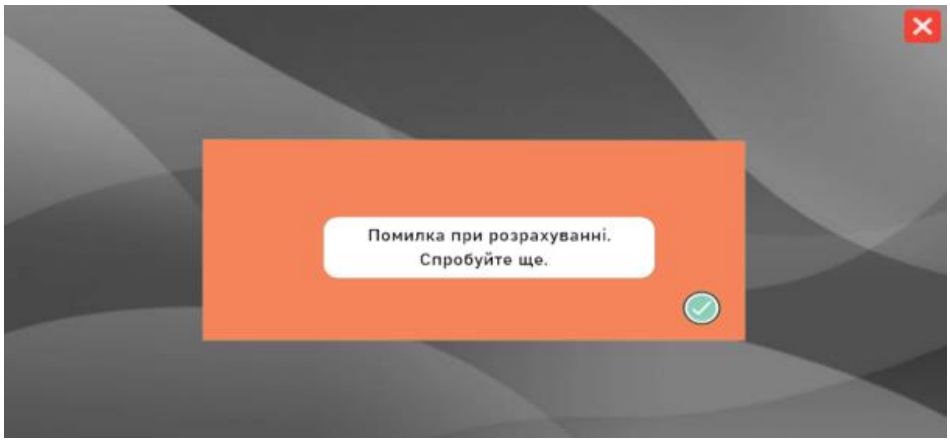

Рисунок 4.9 – Помилка при розрахуванні

У разі успішного виведення результатів розрахунку з'являється вікно з кнопкою «Моделювати розрахунок». Коли користувач натискає на цю кнопку, система надсилає інформацію з розрахунку на симуляцію (візуалізацію), і користувачеві відкривається вікно (рис. 4.10), з уже змодельованими результатами, якщо все пройшло успішно.

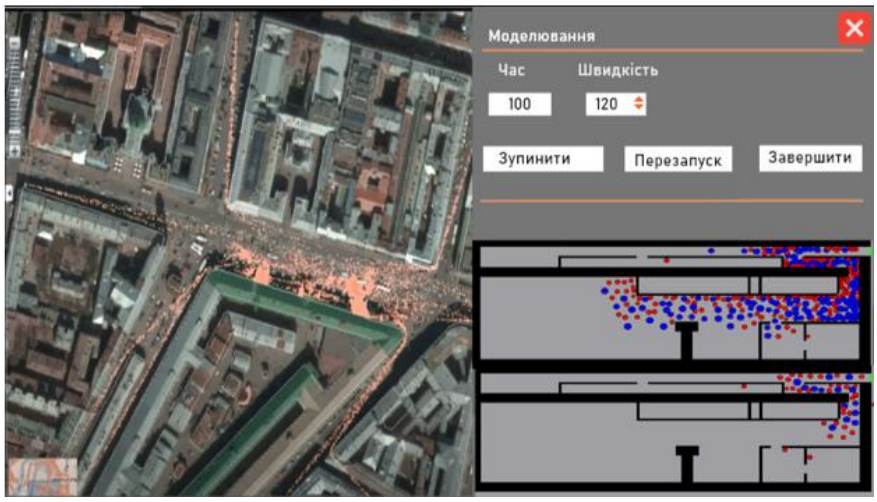

Рисунок 4.10 – Результат моделювання

Якщо результат не вдається змоделювати, з'являється вікно помилки і програма повертає користувача до вікна (рис. 4.11), де він може натиснути кнопку «Моделювати результат», щоб повторити спробу моделювання.

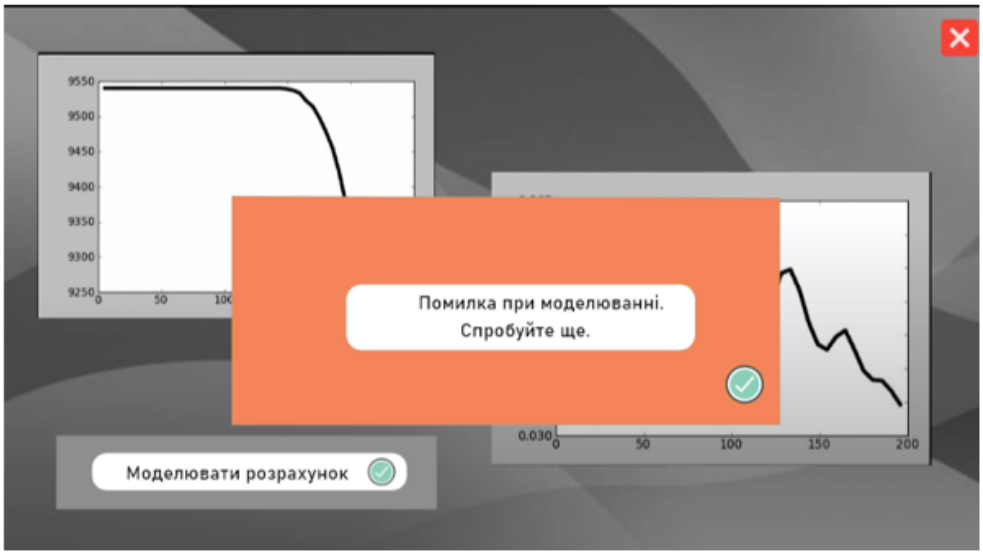

Рисунок 4.11 – Помилка при моделюванні

Система також пропонує додаткові функції. Наприклад, натиснувши на пункт «Налаштування» у спливаючому меню, можна змінити мову та налаштувати режим симуляції (рис. 4.12).

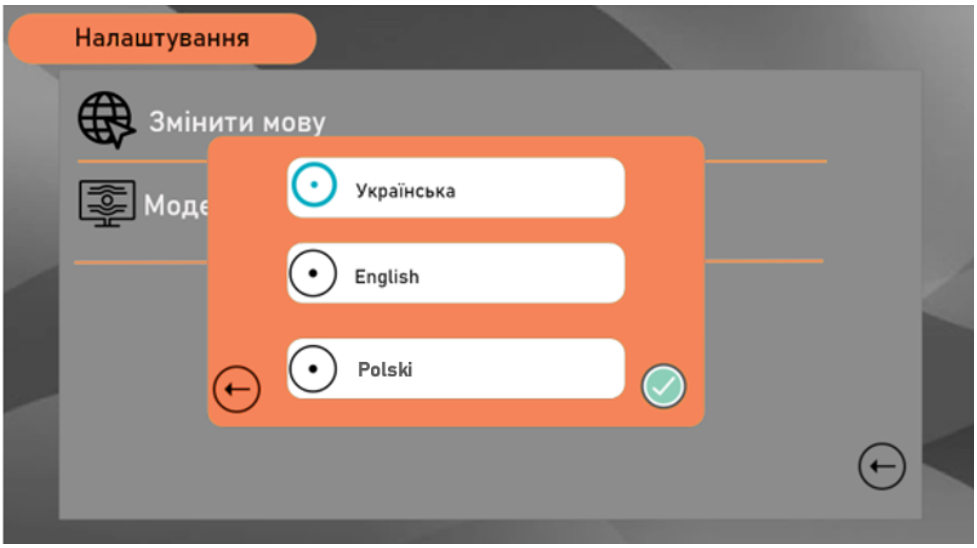

Рисунок 4.12 – Зміна мови

В налаштуванні моделювання (рис. 4.13), є можливість виконати такі дії:

– змінити одиниці виміру;

– змінити швидкість моделювання.

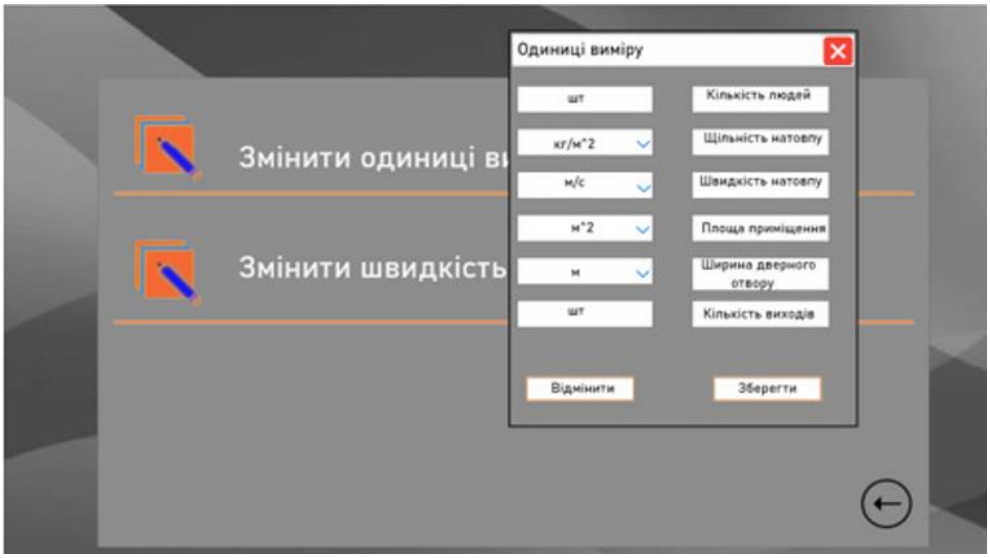

Рисунок 4.13 – Налаштування моделювання

В елементі «Безпека» (рис. 4.14), можна буде виконати наступні дії:

– змінити пароль;

– видалити всі дані;

– видалити обліковий запис.

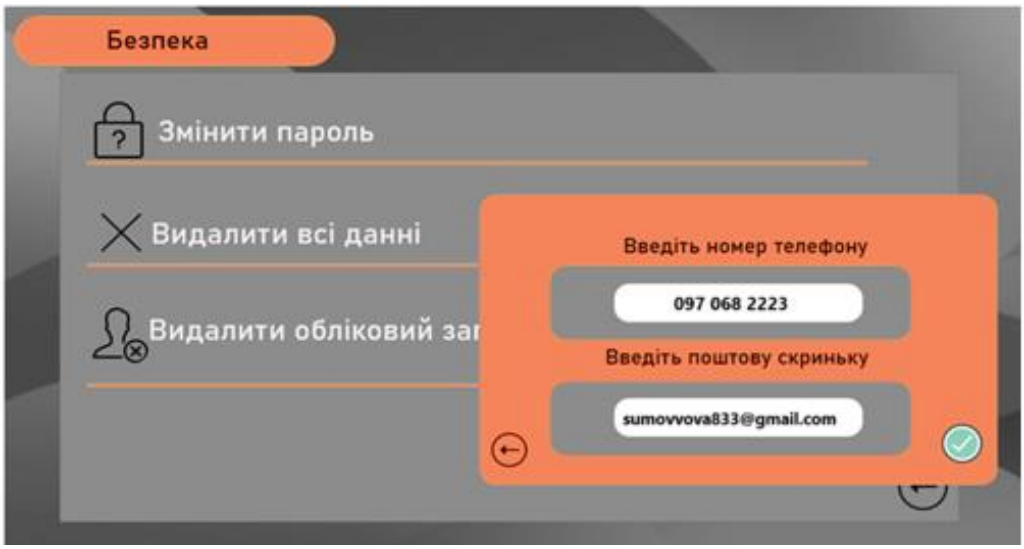

Рисунок 4.14 – Змінити пароль

4.2 Дослідження поведінки людей при змінних умовах

Перший етап – дослідження залежності ефективності евакуації від розміру приміщення та кількості людей.

Мета: Оцінити, як площа приміщення і кількість осіб впливають на час евакуації.

Методологія:

– створити моделі приміщень із різною площею (500, 1000, 1500 кв.м);

– змінювати кількість людей (100, 500, 1000);

– оцінювати час евакуації за допомогою моделі, наведеної в магістерській роботі.

Очікувані результати: Час евакуації лінійно залежить від розміру приміщення за фіксованої щільності.

Збільшення чисельності при однаковій площі суттєво впливає на затримки.

Другий етап – оцінка впливу кількості виходів на швидкість евакуації.

Мета: Дослідити, як збільшення кількості виходів зменшує час евакуації.

Методологія:

– моделювання приміщення із фіксованою площею (наприклад, 1000 кв.м) і щільністю натовпу;

– додавання виходів: від 1 до 5.

Фіксація часу евакуації при кожній зміні.

Очікувані результати: Додаткові виходи суттєво знижують час евакуації, особливо при високій щільності натовпу.

Третій етап – вплив паніки на евакуацію.

Мета: Проаналізувати, як емоційний стан людей впливає на їхню поведінку і час евакуації.

Методологія:

– створити два сценарії: один із впливом паніки, інший — без;

– додати параметри, які враховують хаотичність руху при паніці;

– оцінити, як зміни поведінки впливають на час евакуації.

Очікувані результати: Паніка значно збільшує час евакуації.

Додавання керуючого персоналу знижує вплив паніки на процес.

Четвертий етап – аналіз оптимальних шляхів евакуації.

Мета: Оцінити, як зміни у плануванні приміщення (ширші коридори, розташування виходів) впливають на швидкість евакуації.

Методологія:

– моделювання різних сценаріїв розташування виходів і ширини коридорів;

– аналіз часу евакуації за допомогою моделі;

– порівняння результатів для різних сценаріїв;

– очікувані результати: Оптимізація ширини коридорів та розташування виходів суттєво покращує ефективність.

Графіки, що ілюструють результати дослідження:

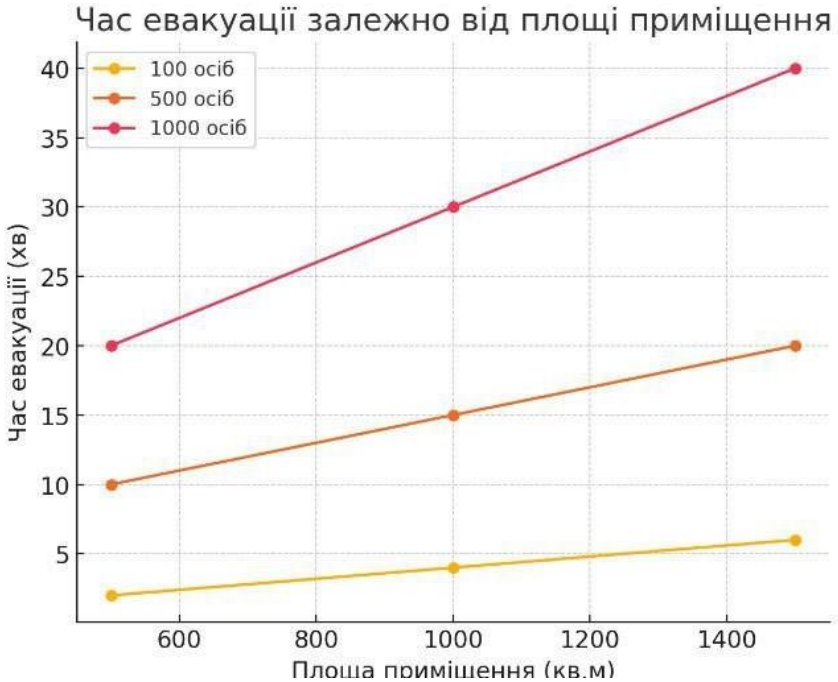

Рисунок 4.15 - Час евакуації залежно від площі приміщення

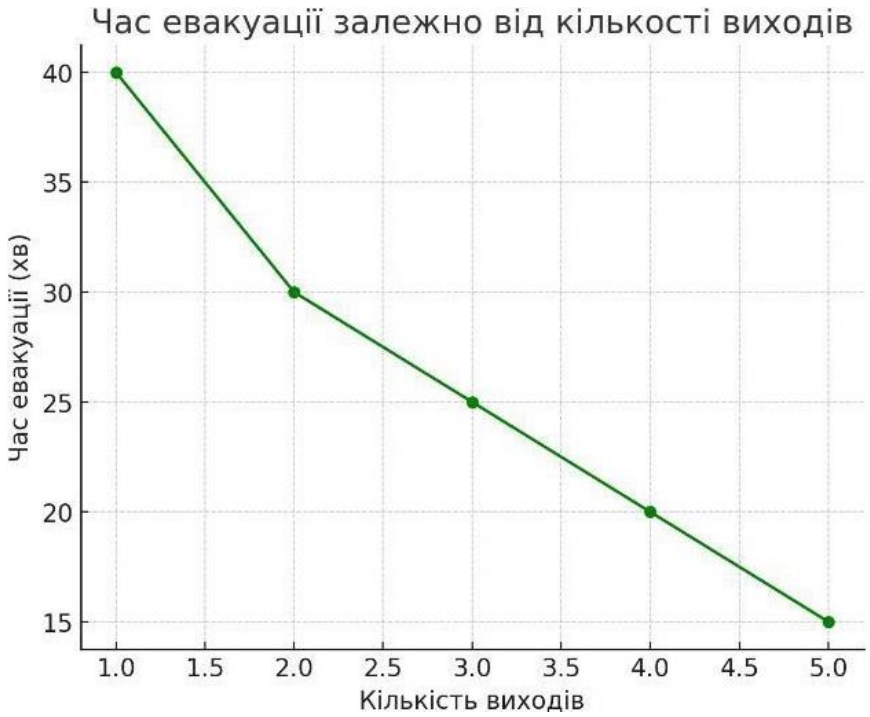

Рисунок 4.16 - Час евакуації залежно від кількості виходів

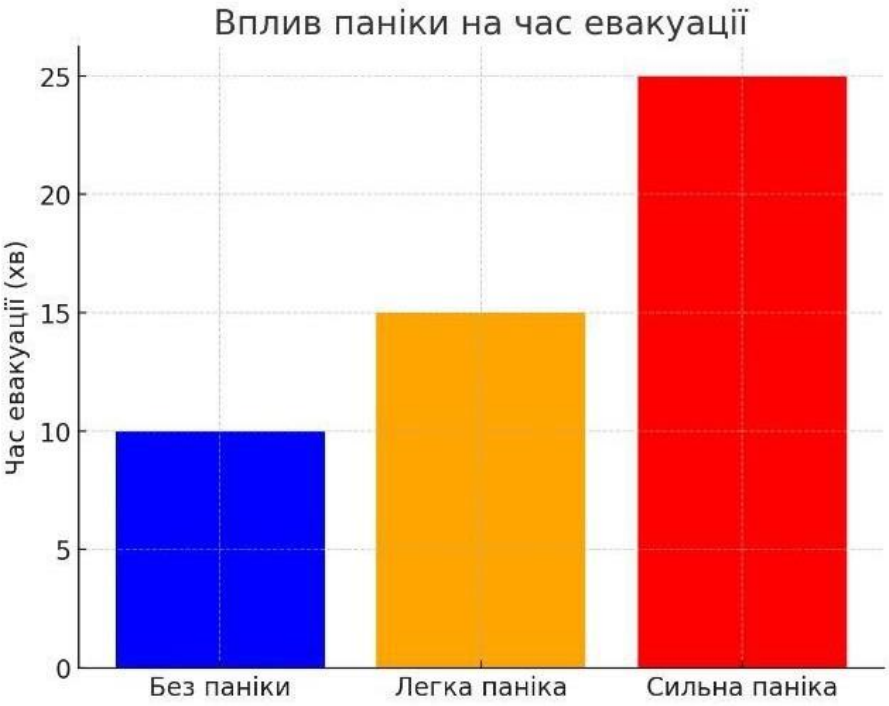

Рисунок 4.17 - Вплив паніки на час евакуації:

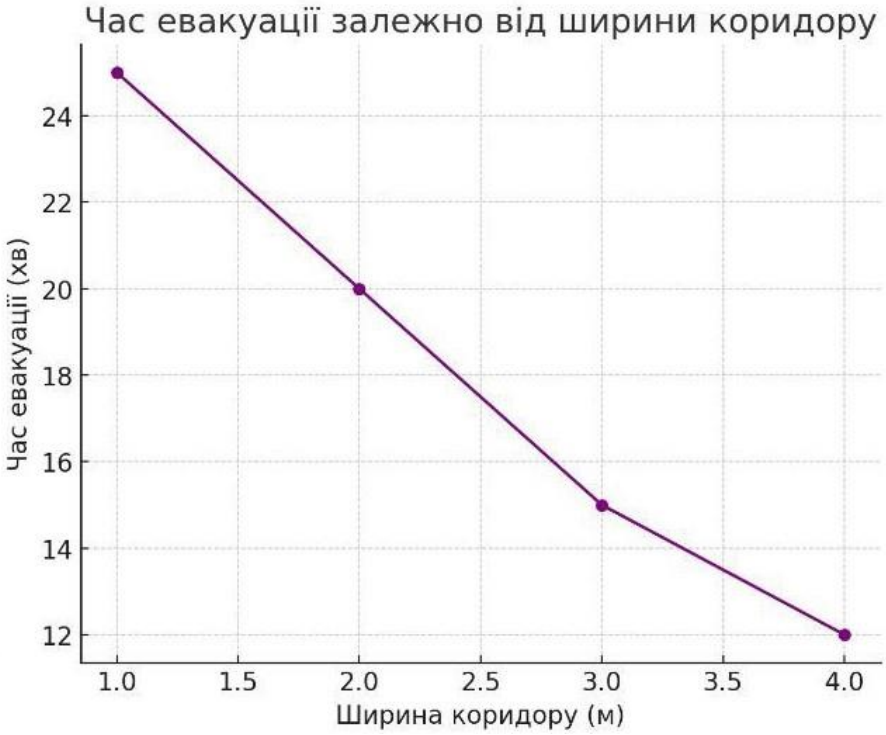

Рисунок 4.18 - Вплив ширини коридору на час евакуації

За результатами проведеного дослідження встанволено, що зі збільшенням площі час евакуації зростає, особливо для великих натовпів (рис. 4.15). Додавання виходів значно скорочує час евакуації, особливо коли їх кількість зростає з 1 до 3. (рис 4.16). У стані сильної паніки час евакуації зростає більш ніж удвічі порівняно зі спокійним станом (рис. 4.17). Збільшення ширини коридору з 1 до 4 метрів суттєво знижує час евакуації (рис. 4.18).

ВИСНОВКИ

В даній магістерській роботі було використано об'єктно-орієнтований підхід до моделювання та уніфіковану мову UML для проєктування комп'ютерної системи для моделювання поведінки людини в екстримальних ситуаціях. Також було створено модель інтерфейсу користувача для подальшої реалізації та тестування.

Теоретична частина магістерської роботи описує інформацію про моделювання та проєктування спеціалізованих комп'ютерних систем, а також тестування інтерфейсу користувача та системи в цілому.

У проєкті були розглянуті наступні питання:

проаналізовано та описано предметні області системи. Визначено вимоги до проєктування та розробки системи;

– спроєктували систему та створили діаграму варіантів використання системи за допомогою кейс-інструменту;

– розглянули сучасні інструменти моделювання, щоб зробити роботу з системою більш продуктивною;

– пояснив етапи реалізації, надав код системи та представив рішення для побудови користувацького інтерфейсу;

– ручне тестування системи.

Проведено дослідження під час якого змінювалися умови що можуть вплинуть на поведінку людини в екстримальних ситуаціях. За результатами якого виявлено, що збільшення площі час евакуації зростає, а додавання додаткових виходів та збільшення ширини коридорів навпаки його зменшує. Також на збільшення часу евакуації впливає емоційний стан людини а саме - паніка.

Результатом даної магістерської роботи є те, що розроблена система може бути використана для автоматизації розрахунків поведінки людини з метою підвищення продуктивності праці та зменшення впливу людського фактору.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Екстримальна психологія [Електронний ресурс] – URL: https://onu.edu.ua/pub/bank/userfiles/files/fpsr/navchalni_materialy/metodychne_zabezpechenya/053_bakalavr/mr_ekstremalna_psykhologiya.pdf

2. Комп'ютерне моделювання: системи і процеси [Електронний ресурс] – URL: https://viduus.net/wp-content/uploads/2019/02/Razrabotka-trebovanij-k-programmnomu-obespecheniyu.pdf

3. Розробка інформація ресурсів та систем [Електронний ресурс] – URL: http://www.its.kpi.ua/itm/tkot/Students/Lec2_2-State-Classes-D.pdf

4. Використання уніфікованої мови візуального моделювання UML (UNIFIED MODELING LANGUAGE) як інструменту підтримки проєктування інформаційних систем [Електронний ресурс] – URL: https://www.lute.lviv.ua/fileadmin/www.lac.lviv.ua/data/DOI/2522-1256-2019-24-16.pdf

5. Застосування UML для моделювання та проєктування інформаційних систем [Електронний ресурс] – URL: https://ela.kpi.ua/server/api/core/bitstreams/eb5f589c-6cb5-4fe6-acff-d88b193777af/content

6. 14.2.3. Інкапсуляція. StudFiles. [Електронний ресурс] – URL: https://studfile.net/preview/9445425/page:51/.

7. Entity relationship diagram (ERD) - what is an ER diagram?. SmartDraw is a Unified Visual Collaboration App | Diagramming, Whiteboarding, and Data Visualization. [Електронний ресурс] – URL: https://www.smartdraw.com/entity-relationship-diagram/.

8. Що таке GUI?. Онлайн-курси від компанії QATestLab | Головна сторінка. [Електронний ресурс] – URL: https://training.qatestlab.com/blog/technical-articles/what-is-graphical-user-interface-design/.

9. Tkinter python: як використовувати. FoxmindEd. [Електронний ресурс] – URL: https://foxminded.ua/tkinter-python-yak-vykorystovuvaty/.

10. Aptukov A. M., Bratsun D. A., Lyushnin A. V. Modeling of behavior of panicked crowd in multi-floor branched space. Computer research and modeling . 2013. Vol. 5, no. 3. P. 491–508.

11. Шумов. Інтернет-конференції НУБіП України. [Електронний ресурс] – URL: http://econference.nubip.edu.ua/index.php/grpi/grpi24/paper/view/3557.